NEW PROBABILISTIC GRAPHICAL MODELS AND META-LEARNING APPROACHES FOR HIERARCHICAL CLASSIFICATION, WITH APPLICATIONS IN BIOINFORMATICS AND AGEING

A THESIS SUBMITTED TO
THE UNIVERSITY OF KENT
IN THE SUBJECT OF COMPUTER SCIENCE
FOR THE DEGREE
OF PHD.

By
Fabio Fabris
September 2017

Abstract

This interdisciplinary work proposes new hierarchical classification algorithms and evaluates them on biological datasets, and specifically on ageing-related datasets. Hierarchical classification is a type of classification task where the classes to be predicted are organized into a hierarchical structure. The focus on ageing is justified by the increasing impact that ageing-related diseases have on the human population and by the increasing amount of freely available ageing-related data.

The main contributions of this thesis are as follows. First, we improve the running time of a previously proposed hierarchical classification algorithm based on an extension of the well-known Naive Bayes classification algorithm. We show that our modification greatly improves the runtime of the hierarchical classification algorithm, maintaining its predictive performance.

We also propose four new hierarchical classification algorithms. The focus on hierarchical classification algorithms and their evaluation on biological data is justified as the class labels of biological data are commonly organized into class hierarchies. Two of our four new hierarchical classification algorithms – the "Hierarchical Dependence Network" (HDN) and the "Hierarchical Dependence Network algorithm based on finding non-Hierarchically related Predictive Classes" (HDN-nHPC) – are based on Dependence Networks, a relatively new type of probabilistic graphical model that has not yet received a lot of attention from the classification community. The other two hierarchical classification algorithms we proposed are hybrid algorithms that use the hierarchical classification models produced by the Predictive Clustering Tree (PCT) algorithm. One of the hybrids combines the models produced by the PCT algorithm and a Local Hierarchical Classification (LHC) algorithm (which basically induces a local model for each class in the hierarchy). The other hybrid combines the models produced by the PCT and HDN algorithms.

We have tested our four proposed algorithms and four other commonly used hierarchical classification algorithms on 42 hierarchical classification datasets. 20

of these datasets were created by us and are freely available for researchers. We have concluded that, for one out of the three hierarchical predictive accuracy measures used in our experiments, one of our four new algorithms (the HDN-nHPC algorithm) outperforms all other seven algorithms in terms of average rank across the 42 hierarchical classification datasets.

We have also proposed the first meta-learning approach for hierarchical classification problems. In meta-learning, each meta-instance represents a dataset, meta-features represent dataset properties, and meta-classes represent the best classification algorithm for the corresponding dataset (meta-instance). Hence, meta-learning techniques for classification use the predictive performance of some candidate classification algorithms in previously tested datasets, and dataset descriptors (the meta-features), to infer the performance of those candidate classification algorithms in new datasets, given the meta-features of those new datasets.

The predictions of our meta-learning system can be used as a guide to choose which hierarchical classification algorithm (out of a set of candidate ones) to use on a new dataset, without the need for time-consuming trial and error experiments with those candidate algorithms. This is particularly important for hierarchical classification problems, as the training time of hierarchical classification algorithms tends to be much greater than the training time of 'flat' classification algorithms. This increased training time is mainly due to the typically much greater number of class labels that annotate the instances of hierarchical classification problems.

We have tested the predictive power of our meta-learning system and interpreted some generated meta-models. We have concluded that our meta-learning system had good predictive performance when compared to other baseline meta-learning approaches. We have also concluded that the meta-rules generated by our meta-learning system were useful to identify dataset characteristics to assist the choice of hierarchical classification algorithm.

Finally, we have reviewed the current practice of applying supervised machine learning (classification and regression) algorithms to study the biology of ageing. This review discusses the main findings of such algorithms, in the context of the ageing biology literature. We have also interpreted some of the hierarchical classification models generated in our experiments. Both the above literature review and the interpretation of some models were performed in collaboration with an ageing expert, in order to extract relevant information for ageing research.

Acknowledgements

First of all, I would like to thank my PhD supervisor, Dr. Alex Alves Freitas for his amazing commitment and dedication to my PhD. His knowledge, diligence and attention to detail never cease to impress me. Alex was not just a great PhD supervisor, he also went out of his way to greatly help me on several issues beyond the scope of my PhD, for which I am very grateful.

Second, I would like to thank the ageing experts Dr. Jennifer M. A. Tullet (University of Kent) and Dr. João Pedro de Magalhães (University of Liverpool) for assisting me with the complex task of interpreting the classification models from a biological point of view. Their insights greatly improved the quality of this work.

Third, I would like to thank my family, specially my fiancé, Carol, who has always supported me with her loving embrace even in the most difficult and stressful times. I also would like to thank my mother, sister and father, who have always encouraged me to pursue my PhD and have always looked after me, even from afar.

I would also like to thank my friends, who made the last three years much more enjoyable. I thank specially Izabela, and her husband Guilherme, who I was very lucky to befriend during my time in the UK, they are among the most amazing people I know. I would also like to thank my great 'chaotic' (they will get it :)) Brazilian friends, who have made my time here more enjoyable with their wit and humor.

I also thank Dr. Flávio Miguel Varejão, my former Brazilian supervisor, who has introduced me to the world of data mining and has started my contacts with Alex. I have worked with Flávio for several years, in many different projects, this gave me great experience in the field of machine learning.

I thank Capes, the Brazilian research agency that has sponsored my PhD, without which this research would be impossible. I thank specially Maria Gabriela Rodrigues, Priscylla Olivo Moreira and Fernanda Botelho de Arruda, who have

answered many questions regarding my scholarship.

Lastly, I also thank the anonymous reviewers from the Bioinformatics journal; the Biogerontology journal; the IEEE/ACM Transactions on Computational Biology and Bioinformatics journal; the 2015 International Conference on Tools with Artificial Intelligence (ICTAI); the 2014 International Workshop on Data Mining in Bioinformatics (BIOKDD); the 2014 IEEE Symposium on Computational Intelligence and Data Mining (SSCI-CIDM); and the 2015 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD). These reviewers have given important feedback about the papers I have submitted during my PhD, their comments undoubtedly improved the quality of my research. I must also thank the members of my supervisory panel: Dr. Peter Rodgers and Dr. Fernando Otero, who have also contributed to my research with their insightful comments.

Contents

List of Tables

List of Figures

Mathematical Notation

$\mathbf{C_{-i}}$ The vector of predictions of the classes in the Markov blanket of C_i.

$\mathbf{C_{cand}}$ The vector containing ordered pair of class variables in the form $(C_i \leftharpoonup C_{i'})$ such that the classes are not descendants (or ancestors) of each other.

$\mathbf{C_i^{CB}}$ The "Class Blanket of C_i", a subset of $\mathbf{C}$, the minimal set of classes that affect the estimation of C_i .

$\mathbf{C}^{\dagger}$ The vector containing ordered pair of class variables in the form $(C_i \leftharpoonup C_{i'})$ such that the estimation given by $P(C_i|\mathbf{x}, C_{i'})$ is significantly different from $P(C_i|\mathbf{x})$.

$\mathbf{C_i^{FB}}$ The "Feature Blanket of C_i", a vector, subset of $\mathbf{x}$, the minimal set of predictive features that affect the estimation of C_i.

C_i The i-th class label of the vector $\mathbf{C}$ that contains the class labels of the hierarchy. In the context of a probability function, a random variable that may take the values c_i and $\tilde{c_i}$, $C_i = c_i$ is the event of the current instance being classified as belonging to the i-th class label and $C_i = \tilde{c_i}$ the event of the current instance being classified as *not* belonging to the i-th class.

$D^{\tilde{c_i}}_{(i,i')}$ **and** $D^{c_i}_{(i,i')}$ Instances are assigned to $D^{\tilde{c_i}}_{(i,i')}$ if $C_{i'} = \tilde{c_i}$, conversely, assigned to dataset $D^{c_i}_{(i,i')}$ if $C_{i'} = c_i$. Additionally, $D^{\tilde{c_i}}_{(i,i')} \cup D^{c_i}_{(i,i')} \equiv D_{learn}$, and D_{learn} is a random partition of 70% of the training set.

$\mathcal{F}$ A function that maps an instance to one or more class labels.

$G = (V, E)$ A directed graph G, defined by a set of vertices V and a set of edges E.

hF Hierarchical F-Measure.

hP Hierarchical Precision.

hR Hierarchical Recall.

i Iterator for class labels.

J The number of instances.

j Iterator for instances.

M The size of vector $\mathbf{x}$, that represents an instance.

$M^{\tilde{c_i}}_{(i,i')}$, $M^{c_i}_{(i,i')}$ **and** $M_{(i,i')}$ Models trained using, respectively, datasets $D^{\tilde{c_i}}_{(i,i')}$, $D^{c_i}_{(i,i')}$ and D_{learn}.

min_inst The minimum number of instances of the least represented class that must exist for a base classifier to be trained for that class.

min_inst_HDN The minimum number of instances required in each cluster in the PCT to train the dependence network.

N Number of class labels in the class hierarchy.

$N_{c'}$ number of instances with value c' for class label C'.

n_feats The number of most relevant features selected by the by the F-test feature selection algorithm.

pa_i The set of parents of the i-th node of the Dependence Network, i.e., the Markov blanket of C_i.

P_j The set of predicted class labels of the j-th instance.

S_{mean} The mean size of the neighborhood of all class nodes.

T_j The set of true class labels of the j-th instance.

$\mathbf{x}^+$ A vector representing the result of applying a feature selection filter on $\mathbf{x}$ and $\mathbf{C_{-i}}$ (treating them as a single, extended predictive vector), generating a new filtered extended predictive feature vector.

$\mathbf{x}$ A numerical vector representing an instance.

$\mathbf{y_i}$ A binary vector representing the classifications of the neighbourhood of the i-th class label.

Z_{pseudo} A pseudo-normalisation constant.

Acronyms

AUPRC Area Under the Precision Recall Curve.

AUROC Area Under the Receiver Operating Characteristic curve.

BAN Bayesian Augmented Naive Bayes.

BN Bayesian Network.

CFS Correlation-based Feature Selection.

DAG Directed Acyclic Graph.

DN Dependence Network.

DTMR Decision Tree Meta-Ranker.

EC Enzyme Commission.

ELHNB Extended Local Hierarchical Naive Bayes.

FD Classification problems with with every instance having a Full Depth class labels.

GNB Gaussian Naive Bayes.

GO Gene Ontology.

GPCR G Protein-Coupled Receptor.

HDN Hierarchical Dependence Network.

HDN-nHPC Hierarchical Dependence Network based on finding non-Hierarchical Predictive Classes.

HDN-PCT The hybrid between the Hierarchical Dependence Network and Predictive Clustering Tree hierarchical classification algorithms.

HMC Hierarchical Multi-label Classification.

KEGG Kyoto Encyclopedia of Genes and Genomes.

LHC Local Hierarchical Classification.

M-ELHNB Modified Extended Local Hierarchical Naive Bayes.

MPO Mammalian Phenotype Ontology.

NB Naive Bayes classification algorithm.

PCT Predictive Clustering Tree.

PCTEN Predictive Clustering Tree Ensemble.

PGM Probabilistic Graphical Model.

PPI Protein-Protein-Interaction.

PR Prior Ranker.

RBF Radial Basis Function kernel of the SVM classifier.

RR Random Ranker.

SPL Classification problems with with every instance having a Single Path of Labels.

SVM Support Vector Machine.

SVMMR Support Vector Machine Meta-Ranker.

TAN Tree Augmented Naive Bayes.

Chapter 1

Introduction

This work involves inter-disciplinary research, at the intersection of the areas of data mining (or machine learning), gene/protein function prediction and the biology of ageing. In terms of data mining, we focus on the classification task, where the algorithm has to predict the value of a class variable for a given instance, based on values of predictor attributes (or features) describing that instance. More precisely, we address a variation of the classification task called hierarchical classification, where the classes to be predicted are structured into a hierarchy of classes (Silla Jr. and Freitas 2011a).

The focus on classification of protein functions is justified by the importance of this type of biological compound in living organisms. Proteins are the building blocks of life, being responsible for building the structure of cells, speeding up essential chemical reactions, signaling messages between cells, building the internal machinery of the cells, and other essential functions. All proteins are formed by chains of amino acids, which fold into a specific spatial conformation. The blueprints of these chains are in the genes of living organisms, and the process of building proteins from genes involves a process of "gene expression" (Alberts et al. 2008).

Broadly speaking, the goals of this research are to develop new algorithms for hierarchical classification and to apply them to the prediction of hierarchical gene/protein functions, including the functions of ageing-related genes/proteins.

The focus on ageing, as an application domain, is motivated by the fact that ageing research is emerging as a field that promises a significant impact on the life of human beings. As the average age of humans has been steadily increasing over the last decades, age-related diseases are more and more common (de Magalhães

2011). The current ageing-related medical research paradigm focuses on specific diseases instead of investigating the ageing process as a whole. An alternative and increasingly popular approach is to research the genetic roots of human ageing, and try to come up with treatments to the ageing process in general, postponing the onset of several age-related diseases simultaneously.

This thesis' focus on ageing is also motivated by the fact that some proteins were shown to be directly involved in ageing (de Magalhães 2011), that is, it was shown that the over-expression or under-expression of certain genes lead to accelerated aging or a slow down of the ageing process. The discovery of ageing-related proteins and their functions can be crucial to the development of treatments for ageing. By targeting proteins that are involved in the ageing process, we could manipulate and potentially even revert this apparently relentless process.

The study of ageing at the proteomic level involves the analysis of significant amounts of data coming from different sources, therefore it is natural to use automated data mining algorithms to find unknown patterns present in this data. For example, algorithms producing interpretable classification models, such as decision tree algorithms, can be used to predict ageing-related classes and provide insights on the biology of ageing, as will be seen later in this thesis.

1.1 The Ageing Problem

Due to technological advances in medicine and healthcare, human longevity has been increasing significantly for decades (de Magalhães 2011). As a result, diseases associated with the "greying population"; such as cancers, heart conditions, and neurodegenerative illnesses; are affecting an increasingly large number of people. Therefore, the pressure to understand, and maybe slow down or ultimately reverse, the ageing process is building up among researches of several areas. Although a lot of progress has been made in recent years to try to explain why do we age and how to potentially slow down this process, our understanding of the biology of ageing is in its infancy. If we succeed in this task, the potential economical benefit of investing on this type of research is clear: it is projected that the economical value of adding 2.2 extra healthy years to the human population is $7.1 trillion dollars over 50 years in the United States alone (Goldman et al. 2013).

Ageing can be defined as an intrinsic, age-related process of loss of viability and increase in vulnerability (Comfort 1964). Although the vast majority of animals are impacted by the ageing process, there are some species that have no apparent senescence process, i.e. not only their mortality rate is constant during their adult life, there is no evident age-related physiological functional decline in their organs (Jones et al. 2014).

Interestingly, studies have found several ageing-related genes in model organisms such as the mouse (*M. musculus*), the fruit fly (*D. melanogaster*), the worm (*C. elegans*), and the yeast (*S. cerevisiae*), which when turned *on* or *off*, considerably affect the lifespan of the organisms. For instance, Friedman and Johnson (1988) showed that by knocking-out the gene *age-1*, associated with the production of *insulin/insulin-like growth factor 1* (IGF1), it was possible to make the roundworm live twice as long. In mice, disruption of the gene *Prop1*, also related to the production of IGF1, may increase their lifespan by 50% (Brown-Borg et al. 1996). These initial findings were products of laborious "wet-lab" experimentation based on fortuitous biological observations.

Recently, the cost of extracting genomic and proteomic data from organisms has decreased many-fold. Researchers now have access to vast public collections of biological data. Many biological databases contain gene or protein sequence information (e.g. the Universal Protein Resource (The Uniprot Consortium 2010)) and possibly a classification of the biological processes where a gene or protein is involved in a curated hierarchical ontology (e.g. the Gene Ontology (GO) (Harris et al. 2004)). The existence of these widely used hierarchical classes justifies the use of hierarchical classification methods to analyse this data.

1.2 Bioinformatics

Bioinformatics is a relatively new discipline that emerged when computer scientists, statisticians and biologists got together to answer biological questions by analysing large quantities of biological data. In particular, the main study object of bioinformatics in the last decades has been genomic data, i.e. the gene sequences of organisms. To appreciate how the study of the genome helps us understand how life works, we need to understand what genes are and what are their functions.

Genes are hereditary discrete units of genetic information discovered by Gregor Mendel in 1859. The actual molecular structure of the DNA, that encodes the genes in nucleotides, was discovered much later, in 1953, by James Watson and Francis Crick (Watson and Crick 1953). Generally speaking, genes contain the instructions necessary to create proteins, which in turn are used in many biological tasks, ranging from structural composition to internal communication in the cell.

Despite their sophistication, all proteins are composed by chains of only 20 types of amino acids. Amino acids are small molecules that serve as the building blocks of proteins. Although proteins are encoded into a linear sequence of amino acids, proteins usually have complex three-dimensional structures. The different physiochemical properties of amino acids mold the final configuration of protein molecules and, consequently, their functions.

Given the gene sequence of an organism, it is tempting to try to discover the function of genes and their respective proteins. The "wet-lab" experimentation to achieve this is usually laborious and often involves introducing mutations in organisms that are probably harmful, which makes it an unfeasible study to be carried out in humans. For these reasons, it is very common to apply sequence alignment tools such as BLAST to find similar sequences that are already annotated and assign those annotations to the unannotated sequence (Madden 2013).

This simple approach works well in some cases, however it has some problems (Freitas and de Carvalho 2007). First, it assumes that similar sequences share similar functions. This is not always true, given that few changes in the amino acid sequence of a protein may result in a very different function for that protein. Second, it ignores hierarchical relations among the classes annotated for the protein. One of the goals of this work is to develop more sophisticated algorithms to annotate unknown-function genes or proteins by considering a hierarchical taxonomy of gene/protein functional classes.

1.3 Hierarchical Classification

This work focuses on the Hierarchical Multi-label Classification (HMC) problem, a type of supervised learning task that naturally emerges in several real-world problems. The HMC problem consists of learning a classification model given a pre-defined class taxonomy and instances annotated with classes from that taxonomy. With the learnt model, we wish to predict which classes of the hierarchy new instances (unseen in the learning phase) belong to. There are notorious successful applications of this technique (Silla Jr. and Freitas 2011a), for instance: in document classification, where it is intuitive to annotate instances in topics that are organised hierarchically; in image classification, where instances are commonly organised in trees; and in bioinformatics, the focus of this work, where functions of genes and proteins are commonly organised in ontologies organised as Directed Acyclic Graphs (DAGs) or as trees. An example of such ontology is the Gene

Ontology (Harris et al. 2004).

In the hierarchical classification setting this thesis focuses on, the classes are organised into an "IS-A" ontology. In such ontologies, the classes (the terms of the ontology) are organised into a tree or DAG (Directed Acyclic Graph), where each directed edge represents a specialisation/generalisation relationship between two terms. For instance, in an ontology describing animals, there would be an edge from *bat* to *flying animal*, as a *bat IS-A flying animal.* In other words, a *bat* is a specialisation of a *flying animal*, or a *flying animal* is a generalisation of a *bat*. Note that in such hierarchy, if an instance is annotated with the term *bat* it is also implicitly annotated with the term *flying animal*, as every *bat* is also a *flying animal*. However, the opposite is not true: if an instance is annotated with the term *flying animal*, it is not necessarily a *bat*.

More formally, a class taxonomy is a set of relationships defined over a partially ordered set $(C, \prec)$ where C enumerates all classes under consideration and $\prec$ represents the "IS-A" relationship. Intuitively, the relationship must be rooted, asymmetric, anti-reflexive and transitive, formally defined as (Silla Jr. and Freitas 2011a):

- Rooted condition: $\exists\ root \in C \mid \forall\ C_i \in C, (C_i \neq root) \rightarrow (C_i \prec root)$,
- Asymmetric condition: $\forall\ C_i, C_{i'} \in C, (C_i \prec C_{i'}) \rightarrow (C_{i'} \not\prec C_i)$,
- Anti-reflexive condition: $\forall\ C_i \in C, (C_i \not\prec C_i)$,
- Transitive condition: $\forall\ C_i, C_{i'}, C_{i''} \in C, (C_i \prec C_{i'}), (C_{i'} \prec C_{i''}) \rightarrow (C_i \prec C_{i''})$.

Although using traditional (*flat*) classifiers for hierarchical classification is possible, the literature in the area consistently shows that algorithms specifically designed for HMC outperform the naive approach of treating the problem as a *flat* multi-label classification problem (Silla Jr. and Freitas 2011a), justifying the effort of developing new algorithms for hierarchical classification.

1.4 Hierarchical Classification Algorithms Based on Dependence Networks

Probabilistic Graphical Models (PGMs) are a useful way to represent real-world probabilistic data. A PGM has two components: its structure and a set of Conditional Probability Distributions (CPDs). The representation and meaning of the structures of PGMs differ depending on the type of PGM used (Koller and Friedman 2009). In Bayesian Networks (BNs), for instance, the structure is a DAG, whose edges represent potentially causal relationships between the nodes (the random variables). The second component of PGMs, the set of CPDs, models the probability distribution of each random variable present in the structure, given the value of its parent random variables (the parents of a random variable C is the set of random variables with an outgoing edge pointing to C).

A Dependence Network (DN) is a relatively new type of PGM that has the advantage of having a more flexible structural definition than other types of PGMs such as BNs. DNs allow for cycles in the structural representation of the model, and the edges in a DN do not model *causal* relationships between random variables. Instead, the set of parents of a random variable in a DN represent the Markov blanket of a random variable. The Markov blanket of a random variable C is the minimal set of other variables that make the estimation of the probability distribution of C independent from the value of all other random variables (Heckerman et al. 2001).

Note that in BNs the parents of a random variable are *not* the variable's Markov blanket. That is, in BNs unconnected random variables can affect each other directly. This can be confusing to untrained users of BNs. DNs avoid this possible source of confusion by connecting the random variables that affect each other directly.

We have developed, for the first time (Fabris and Freitas 2014b), hierarchical classification algorithms based on DNs. The first version of our Hierarchical DN algorithm (called simply "HDN") uses expert knowledge present in the definition of the class ontology to model the relationships between classes. We have also developed a second version of the HDN algorithm called HDN based on finding non-Hierarchically related Predictive Classes (HDN-nHPC). This version uses two sources of information to build the structure of DN: expert knowledge (as the HDN algorithm) and new relationships among classes (i.e. relationships not included in the pre-defined class hierarchy) that are automatically detected by the algorithm in a data-driven way. Finally, we have also developed another hierarchical classification algorithm, which is a hybrid algorithm, combining our HDN algorithm with the well-known Predictive Clustering Tree (PCT) algorithm for hierarchical classification (Blockeel et al. 2002). A detailed description of all these new hierarchical classification algorithms based on dependence networks will be given in Chapter 5.

1.5 Meta-Learning

The 'traditional' way of applying classification algorithms to new datasets is to use tacit knowledge of classification specialists to select a subset of promising algorithms. This knowledge comes from the past experiences of the specialist when dealing with similar problems. With this subset of promising algorithms, the specialist runs exploratory experiments, which usually take a substantial amount of time, to decide the most appropriated algorithms for a particular task.

Meta-learning is an advanced research area in the field of data mining (or machine learning). There are different types of meta-learning problems (Brazdil et al. 2008); but in this thesis we focus only on the problem of meta-learning for algorithm recommendation (arguably the most well-known type of meta-learning), hereafter denoted simply meta-learning, for short. More precisely, meta-learning methods aim to automate the process of selecting the best classification algorithm for a given input dataset. They do so by using a machine learning method to find patterns relating the predictive performances of a set of candidate classification algorithms on the current classification dataset (the testing meta-instance) with past performance of that set of candidate classification algorithms on other classification datasets (the training meta-instances). The benefits of this approach is that it is a more systematic (less "ad-hoc") practice; if one uses meta-learners producing interpretable meta-models, useful insights about why some types of classification algorithms are more suitable for some kinds of classification datasets can be derived; and its application is not so dependent on the knowledge of experts, which are often unavailable.

Note that meta-learning is part of what is currently referred to Automated Machine Learning (AutoML). AutoML aims to automate the whole machine learning workflow, including data preprocessing, feature selection, classification algorithm selection and parameter optimisation, model post-processing, and the analysis of results. This goal is very broad, requiring several different techniques to achieve the final objective of automating the whole data mining process. In this thesis we focus only on the task of classification algorithm selection, which has a much smaller scope.

It should be noted that, in general, meta-learning research so far has focused on recommending flat (non-hierarchical) classification algorithms. This thesis is the first work to propose meta-learning for recommending hierarchical classification algorithms, as will be discussed in Chapter 7.

1.6 Objectives and Original Contributions

This interdisciplinary thesis has the overall objectives of: 1) proposing new hierarchical classification algorithms, evaluating their predictive performance across a number of hierarchical classification datasets; 2) proposing a new meta-learning approach for recommending the best hierarchical classification algorithm (out of a set of candidate algorithms) for a given input hierarchical classification dataset; 3) interpreting some hierarchical classification models produced by applying hierarchical classification algorithms to datasets of ageing-related genes/proteins.

While accomplishing these objectives, this thesis presents original contributions to two areas, namely hierarchical classification, a type of data mining (or machine learning) task, and the bioinformatics of ageing. More precisely, the thesis presents the following contributions:

First, we have proposed a modified version of a hierarchical classification algorithm based on an extension of Naive Bayes, and have shown, in experiments with 18 hierarchical classification datasets, that in general the modified algorithm has a runtime substantially shorter than the original algorithm, without sacrificing predictive performance. This contribution is presented in Chapter 4.

Second, we have created 20 new hierarchical classification datasets of ageing-related genes/proteins, varying the species to which the genes/proteins belong, the types of gene/protein properties used as predictive features, and the types of hierarchical classes. We also proposed a new type of predictive feature that can be used for predicting gene/protein functions, based on quantifying the influence of a gene/protein in a KEGG pathway (a well-known type of biological pathway), and used this new feature type in some of our datasets. These created datasets, which are described in Chapter 6, are freely available to other researchers at https://dx.doi.org/10.13140/RG.2.2.34027.23843.

Third, we have proposed four novel hierarchical classification algorithms. Two are based on dependence networks. One is a hybrid between one of our dependence network algorithms and a well-known hierarchical classification algorithm based on decision trees, the PCT algorithm (Blockeel et al. 2002). The fourth new algorithm is a hybrid between PCT and a Local Hierarchical Classification (LHC) algorithm – which essentially builds a local model for each class in the hierarchy. These new algorithms are described in Chapter 5, and they are evaluated by comparing them with several strong baseline algorithms across 42 datasets of ageing-related genes/proteins in Chapter 6. This evaluation also included a statistical analysis of the differences of predictive performances between the proposed algorithms and the baseline ones.

Fourth, we have proposed a new meta-learning approach for selecting, among a set of candidate hierarchical classification algorithms, the most recommended one (i.e. the candidate algorithm which is predicted to have the highest predictive performance) for a given input hierarchical classification dataset. This approach was evaluated in experiments with four candidate hierarchical classification algorithms and two types of classification algorithms (a decision tree and an SVM algorithm) used as meta-learners, which were compared against two simpler baseline meta-learners. In addition, some produced meta-classification models were interpreted to detect patterns about which characteristics of hierarchical classification datasets (described by meta-features) lead to the recommendation of one type of hierarchical classification algorithm over another. This meta-learning approach – which is the first study on meta-learning for hierarchical classification algorithm recommendation – is described and evaluated in Chapter 7.

Fifth, we have interpreted some hierarchical classification models produced by applying hierarchical classification algorithms to our created datasets of ageing-related genes, in the context of the literature on the biology of ageing. This interpretation was performed in collaboration with an ageing biology expert, Dr. Jennifer M.A. Tullet (School of Biosciences, University of Kent), and was reported in Chapter 6. In addition, in order to get further insight on related work on the results of applying data mining methods to ageing-related data, we have also performed a detailed review of the literature on this topic, in collaboration with another ageing biology expert, Dr. João Pedro de Magalhães (Institute of Integrative Biology, University of Liverpool), as reported in Section 3.3.

1.7 Structure of the Thesis

Chapter 2 presents background on data mining concepts and methods relevant for the remainder of the thesis. More precisely, this chapter reviews concepts and methods from the areas of conventional, flat (non-hierarchical) classification, hierarchical classification (the data mining task which is the focus of this thesis), and probabilistic graphical models – including dependence networks, the type of method used as the basis for three of our proposed hierarchical classification algorithms (presented in Chapter 5). This chapter also includes a discussion of related work on exploring class label dependencies in hierarchical classification. We also review the baseline hierarchical classification algorithms that we will compare against our algorithms.

Chapter 3 presents background on ageing and bioinformatics concepts relevant for the remainder of the thesis. More precisely, this chapter briefly reviews two major hierarchical taxonomies of gene/protein functions (which are used as hierarchical classes in our experiments) and the main theories of ageing. This chapter also contains an extensive review of papers interpreting the models generated by applying supervised machine learning (classification and regression) algorithms to ageing-related data.

Chapter 4 first reviews the ELHNB algorithm, which is a version of the Naive Bayes algorithm for hierarchical classification (Merschmann and Freitas 2013). Then, we describe how we modified that algorithm to improve its runtime on a specific type of hierarchical classification problem. We report the results of experiments showing that, in general, the modified algorithm (called M-ELHNB) has runtimes much smaller than the runtimes of the original ELHNB algorithm (particularly on the largest hierarchical classification datasets), whilst both algorithms obtain in general statistically equivalent predictive performances.

Chapter 5 proposes three novel hierarchical classification algorithms based on dependence networks, a type of probabilistic graphical model, as mentioned earlier. Two of these algorithms are entirely based on dependence networks. The other proposed hierarchical classification algorithm is a hybrid between one of the proposed dependence network algorithms and a well-known hierarchical classification algorithm based on decision trees, called the PCT algorithm (Blockeel et al.

2002). This chapter also presents a novel hybrid between the PCT algorithm and a local hierarchical classification algorithm.

Chapter 6 presents the results of the experimental evaluation of the hierarchical classification algorithms proposed in Chapter 5. More precisely, this chapter presents the characteristics of the hierarchical classification datasets that we collected and created; the statistical analysis of the predictive performance results; the interpretation of some classification models with the objective of gaining new biological insight about the ageing process; and finally a time complexity analysis of the most successful (in terms of predictive performance) of the four proposed hierarchical classification algorithms, namely the HDN-nHPC algorithm.

Chapter 7 presents our proposed meta-learning approach for recommending the best hierarchical classification algorithm for a given input dataset, among some candidate hierarchical classification algorithms. The set of candidate algorithms included our proposed HDN-nHPC algorithm and three other algorithms, PCT, PCTEN (PCT Ensemble), and LHC. These four algorithms were described in Chapter 5. This chapter proposes new meta-features for meta-learning in hierarchical classification, and also proposes a new algorithm that divides existing hierarchical classification datasets into a much larger number of hierarchical datasets, in order to increase the number of meta-instances for meta-learning purposes – which is necessary in order to produce more reliable meta-classification models. This chapter also reports the predictive performance results of the meta-learning methods and analyses some of the produced meta-models, in order to try to get insight about which characteristics of hierarchical classification datasets (meta-features) seem to favour the choice of one hierarchical classification algorithm over another.

Chapter 8 presents the conclusions of the thesis. This chapter consists of two parts, namely a summary of the thesis' contributions and several lines for future research.

1.8 Publications Derived from this Research

The research presented in this thesis has resulted in the following publications:

Papers Published in Journals

1. Fabris, F. and Freitas, A. A. (2016). New KEGG pathway-based interpretable features for classifying ageing-related mouse proteins. *Bioinformatics*, 32(19), pp. 2988–2995, DOI: 10.1093/bioinformatics/btw363.

2. Fabris, F., Freitas, A. A. and Tullet, J. (2015). An Extensive Empirical Comparison of Probabilistic Hierarchical Classifiers in Datasets of Ageing-Related Genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(6), pp. 1045–1058, DOI: 10.1109/TCBB.2015.2505288.

Papers Published in the Proceedings of International Conferences

1. Fabris, F. and Freitas, A. A. (2015). A Novel Extended Hierarchical Dependence Network Method Based on non-Hierarchical Predictive Classes and Applications to Ageing-Related Data. In *Proceedings of the 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI'15)*, IEEE, pp. 294–301.

2. Fabris, F. and Freitas, A. A. (2014a). An Efficient Algorithm for Hierarchical Classification of Protein and Gene Functions. In *Proceedings of the 2014 25th International Workshop on Database and Expert Systems Applications (DEXA'14)*, pp. 64–68.

3. Fabris, F. and Freitas, A. A. (2014b). Dependency Network Methods for Hierarchical Multi-label Classification of Gene Functions. In *Proceedings of the 2014 IEEE International Conference on Computational Intelligence and Data Mining*, pp. 241–248.

Submitted Papers Awaiting Review

1. Fabris, F. and Freitas, A. A. (2017). Meta-learning for Hierarchical Classification Applied to Bioinformatics. Submitted to ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

2. Fabris, F., de Magalhães, J. P. and Freitas, A. A. (2017). A Review of Supervised Machine Learning Applied in Ageing Research. Submitted to Biogerontology.

Chapter 2

Background on Data Mining

2.1 Classification

Classification algorithms are a common way to use available data to build tools to help automating complex decision-making processes. These algorithms learn, using existing data, models that classify objects from the problem domain (instances) into categories (class labels), given some attributes that describe those instances.

In this context, we call the data that the algorithm uses to learn the model the *training dataset*. This dataset comprises *instances*, which in turn, are composed of two parts: the first part is a fixed-sized vector that describes the instances, and each element of this vector is called a *feature* or *predictive attribute*. Features are usually encoded as numerical or nominal values, depending on the instance being represented. The second part of an instance is a class variable taking a nominal value (in 'standard' single-label classification) or a vector containing nominal values, representing a set of *class labels* that annotate the instance, in the case of 'non-standard', multi-label classification problems. A class label represents the membership information for a particular *class*.

More formally speaking, classification is the computational task of deriving a function $\mathcal{F} : \mathbf{x} \to C$ from a set of instances – called training instances – in the form $(\mathbf{x_j}, \mathbf{c_j})$, where $\mathbf{x_j}$ are the predictive attributes (or features) of the j-th instance and $\mathbf{c_j}$ is (are) the class(es) of the j-th instance. We call the derivation of $\mathcal{F}$ as the "training phase". After the training phase, we use $\mathcal{F}$ to predict the classes of a set of previously unseen instances given only their predictive attributes (prediction phase). Because there are many approaches to build different $\mathcal{F}$ functions and none is better than all the others in every occasion, it is wise to estimate the predictive performance using some measure, such as the F-Measure (Kiritchenko, Matwin and Famili 2005), to select the best classifier for a given classification task.

Classification problems can be divided in three broad types, namely, *binary classification problems*, *multi-class classification problems*, and *multi-label classification problems*. In binary classification problems each instance is annotated with one out of two class labels. For example, a classification problem to decide if a protein (given numerical values that describe it) is ageing-related or not ageing-related.

In multi-class classification problems the instances are annotated with exactly one class label out of a set of more than two possible class labels. For example, a classification problem to decide which species an individual animal (the instance) belongs to, given a set of pre-determined species and numerical morphological features that describe the animal.

Finally, in multi-label classification problems, each instance can be annotated with more than one (or even none) class labels. This is considered the most challenging type of classification problem, since the classifier has to choose which subset of labels to assign to a new instance, and the number of candidate label subsets is 2^N, where N is the number of labels. An example of a multi-label classification problem is to decide the moods that should be assigned to a song, using a set of pre-defined moods, given a numerical vector that defines the song. Since a song may have multiple moods, this classification problem is a multi-label classification problem.

In this thesis we are particularly interested in a sub-type of multi-label classification problems called hierarchical classification problems, where the multiple class labels are organised into a hierarchy in the form a tree or a DAG (Directed Acyclic Graph). Hierarchical classification will be discussed in detail in Section 2.2.

Naturally, one of the main challenges of applying classification algorithms to real-world problems is how to define the predictive attributes. Two very important processes involved in defining the features are called *feature extraction* (Liu and Motoda 1998), where new features are derived from raw data, and *feature selection* (Liu and Motoda 1998), where a subset of features with high quality is selected from existing ones.

These processes are necessary because most classification algorithms require features with higher discriminative power than the original, raw, data. Furthermore, many classification algorithms do not deal well with uninformative features and uninformative predictive classes, generating models with poor discriminative power.

The performance of classification algorithms are usually estimated using a *testing set*, which was not used during the training phase. Another well-known way of estimating performance is to use a k-fold cross-validation procedure (Witten and Frank 2000), where first the available dataset is divided into k folds of approximately the same size. Next, each fold is temporarily removed from the dataset, one at a time, then the dataset with k-1 folds is used for training, and the held-out fold used for testing. When k is the size of the dataset, this procedure is called *leave-one-out validation*. In each testing step, the prediction of the classification model is compared with the actual class labels and some measure of predictive performance is returned. There are several measures of predictive performance, each one with different biases, and measures suitable for hierarchical classification (the focus of this thesis) will be discussed later.

Among the large number of types of classification algorithms available in the literature (Witten and Frank 2000), we present an overview of the three types of algorithm most relevant for this thesis, namely: decision trees, SVMs and bagging (a type of ensemble method).

2.1.1 Decision Tree Classification Algorithms

In this thesis we have used variations of decision tree classification algorithms. These algorithms are simple and widely used in the data mining community. Decision tree classification algorithms build a decision tree, which is a hierarchical structure composed of *nodes* and *edges*. There are three types of nodes: 1) a *root node*, which is a special node without incoming edges (there is exactly one root node per decision tree), 2) internal nodes, which have exactly one incoming edge and two or more outgoing edges, and 3) leaf nodes, which have exactly one incoming edge and no outgoing edges.

The non-leaf nodes of a decision tree represent attribute test conditions and each outgoing edge represents a possible result of the test condition for the associated non-leaf node. The set of outgoing edges of a node represent all possible (mutually exclusive) results of the condition in the non-leaf node, that is, an instance presented to the non-leaf node will always be associated to one, and exactly one outgoing edge. Each edge in the set of outgoing edges of a non-leaf node leads to either another non-leaf node (with another condition) or a leaf node, where a prediction is made. Figure 2.1 shows these types of nodes graphically.

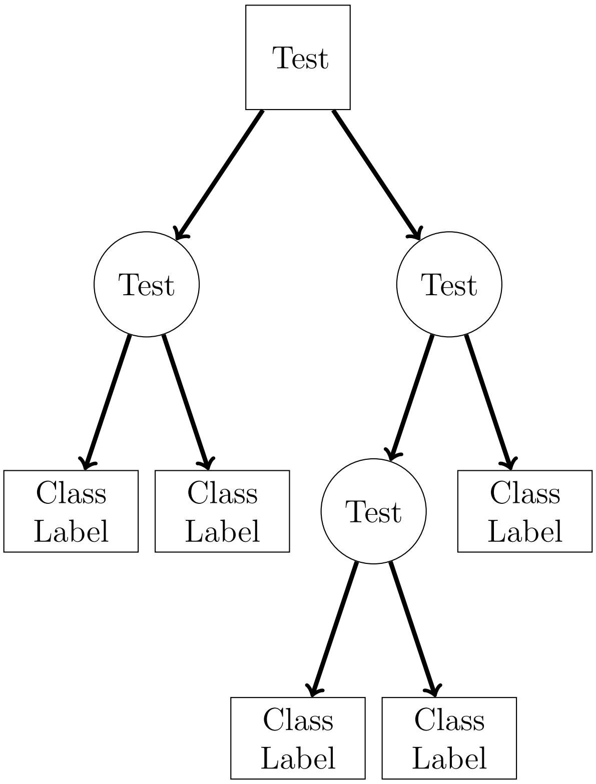


Figure 2.1: Example of a possible decision tree structure. Each non-leaf node (squares and circles) contains a test condition and two outgoing edges, each edge with a possible test result. Note that the root node (the square) does not have incoming edges. Leaf nodes (rectangles) contain the class labels that will be predicted when all test conditions leading to the leaf node are satisfied.

In the testing phase, an instance is presented to the root node of a decision tree, where a condition using one or more (normally one) of the instance's attributes is tested. The outcoming edge associated with the result of the test is identified, and the classification procedure analyses the node that that edges points to. If this node is a non-leaf node, the classification procedure recurses in that node. If the node is a leaf node, the class label associated with the leaf node is returned as the prediction of the decision tree. We call the set of test conditions leading to leaf node and that leaf node's prediction a *rule*. This rule is of the form:

```
IF (test conditions are satisfied by a test instance)
   THEN (assign the class predicted by the leaf node to the instance)
```

As we can see, it is easy to understand *how* a decision tree classifies an instance and, more importantly, *why* a decision tree annotates an instance with a particular class label: one just needs to retrieve the sequence of conditions that were (was) satisfied by the instance. Of course, the decision tree does not tell us the reasons why that particular set of conditions is a good predictor for a class label, but it offers the opportunity for experts to try to interpret the rule in a higher level-domain.

For instance, if the decision tree algorithm finds that interactions with a set of proteins is a good ageing predictor, a biologist expert could analyse these proteins and investigate if they are present in some, yet unknown ageing-related pathway, thus, reaching new biological conclusions.

The most challenging aspect of using a decision tree is the construction of the decision tree itself. Since the number of all valid tree configurations is very large, normally, it is not feasible to test all valid tree configurations to find the optimal decision tree. Therefore, the traditional way of building decision tree algorithms is to use a non-optimal *greedy* algorithm. A greedy algorithm is an optimisation heuristic that always makes local-optimum decisions, without backtracking to previous decisions, until no (significant) improvement can be made.

Most decision tree building algorithms are broadly based on *Hunt's Algorithm* (Hunt, Marin and Stone 1966). This algorithm uses some strategy to find a test condition that partitions the dataset in the current tree node into data subsets, creating a new child node in the decision tree for each subset. Then, the algorithm recurses in each of the child nodes if some measure of quality is not satisfied. If the measure is satisfied, a leaf node is created and the majority class in the subset of instances is used as the prediction of that leaf node.

Therefore, to use Hunt's algorithm to build decision trees one needs to define two procedures: 1) some way to find a good test condition to split the data (sub)set under consideration. This includes some way to create test conditions and some quality measure to evaluate the goodness of each candidate test condition. 2) a criterion(a) for deciding when to stop recursing in the child node created by the test conditions.

Broadly speaking, the data split that a test condition generates can be either binary or multi-way. Multi-way splits can be applied to partition the data into more than two subsets, but can have a much larger search space than binary splits, which always partition the data in two subsets. The decision trees induced in this work only contain binary splits, therefore we focus on this type of split in the next paragraphs.

For binary attributes, binary splits are mandatory, but for nominal and continuous attributes, some strategy has to be applied to create the binary test conditions (Quinlan 1993; Rokach and Maimon 2005). Usually, for continuous attributes, binary splits are created by sorting all unique numerical values in the attribute's domain and using the mean value of two consecutive values in the sorted list as a candidate split condition using the "greater than" and "smaller than" operators.

For non-ordinal nominal attributes, every possible binary grouping is tested. Note that the total number of distinct binary divisions of k elements is $2^k - 1$. For this reason, the number of distinct non-ordinal nominal attributes should be constrained to a small number to keep the running time reasonable. If the nominal attributes are ordinal, it is desirable to use the *greater-or-equal-than* and *smaller-than* operators in a way analogous to continuous attributes, instead of ignoring the element ordering and dividing the values into arbitrary groups. The approach of considering element ordering is less computationally expensive and uses domain knowledge in a more effective way.

The decision tree induction algorithm tests all possible binary splits for all features and uses some measure of partition quality to choose the best split. The basic idea of such quality measure is to reward data partitions where in each data subset the large majority of (ideally all) instances belongs to a single class, and different data subsets are associated with different classes. Commonly used quality measures formalising this basic idea are the *Entropy*, *Gini Index*, and *Classification Error*, for details refer to (Tan, Steinbach and Kumar 2006; Rokach and Maimon 2005).

In addition, commonly, the decision tree inducing algorithm has a user-specific criterion(a) to decide if the algorithm should keep recursing on the subsets generated by the test conditions. That is, the user must inform the decision tree induction algorithm when it should stopping trying to improve the quality of decision tree.

This is necessary for two reasons: first to reduce the decision tree size, which facilitates the tree's interpretation by users, which can be very difficult if very deep decision trees are induced. Second, and more importantly, to try to avoid model over-fitting, that is, to avoid the creation of deep decision trees that classify the training instances very well, but fail to predict training instances with reasonable predictive accuracy.

Next, we discuss the SVM classification algorithm, which was also extensively used in this thesis.

2.1.2 Support Vector Machines (SVMs)

Another type of classification algorithm that has been extensively used in this thesis is Support Vector Machine (SVM) algorithms. Like decision tree algorithms, SVM is a popular type of classification algorithm, but unlike decision tree algorithms, it has a complex mathematical derivation and generates classification models that are in general, too abstract and difficult to interpret. This increased model complexity has the payoff of (commonly) increased predictive performance.

A linear SVM classification algorithm finds a hyperplane (a plane, where the number of dimensions is the number of features) that best separates the data into two classes. In other words, SVM finds decision boundaries with the largest possible margin between the hyperplane and the closest instances (called 'support' instances). It has been shown that large margin models minimise the generalisation error of the SVM classification algorithm (Chapelle et al. 2002).

The procedure to find this hyperplane involves solving a quadratic optimisation problem, which has a complexity of $O(n^3)$, where n is the number of instances (Bordes et al. 2005). Note that unlike most decision tree algorithms, the SVM algorithm finds the optimal separation boundary between classes. This difference is key to explain why the SVM algorithms have, normally, better predictive performance than decision tree algorithms and other classification algorithms.

Note that our discussion so far has only referred to linear SVMs, which are normally not well suited to problems with non-linear relations between features and classes, as is the case with most real-world classification problems.

To solve this, it has been proposed to explicitly map the original non-linearly separable problem to another higher dimension space (Boser, Guyon and Vapnik 1992), where the problem has a linear separation, and use the SVM to find a linear separation in the transformed space, which is actually a non-linear separation in the original space.

However, performing this transformation explicitly is often very computationally expensive and it is subjected to the phenomena know as "the curse of dimensionality", which basically states that the predictive performance of classification algorithms decreases as the number of dimensions increases (Bengio, Delalleau and Le Roux 2005).

Fortunately, a mathematical technique known as the "kernel trick" (Boser, Guyon and Vapnik 1992) has been devised to solve these problems. By using this technique, it is not necessary to explicitly map the instances to the higher dimensional space. An implicit mapping can be performed, which is both more computational efficient and avoids the “curse of dimensionality”. The user of the SVM algorithm needs only to define a *kernel function*, which is a function that maps a pair of instances to a numerical value that represents the distance between the instances in the transformed space. These kernel functions usually have parameters that also need to be chosen by the user of the algorithm.

The most basic version of the SVM algorithm is very sensitive to outliers (instances with very unusual feature values). Because the quadratic optimisation algorithm always tries to find the optimal decision boundary that perfectly separates the instances, a single outlier may completely change the parameters that define the separation hyperplane. For this reason, a *slack* variable was introduced in the formulation of SVM algorithms (Cortes and Vapnik 1995). This user-defined variable controls how tolerant the SVM algorithm should be to outliers, eventually allowing for a few classification errors if a wider margin can be achieved.

In summary, the benefits of SVM algorithms are that: 1) it finds the optimal solution of a well-defined optimisation problem, instead of using greedy local-search heuristics, 2) it is capable of mapping the original classification problem to another space, which may greatly improve the predictive performance of the algorithms.

Some disadvantages of SVM algorithms include: 1) the difficulty of interpreting the generated model, 2) the definition of the slack parameter and the choice of the best kernel function, which are difficult to be estimated by the user, 3) the original derivation only deals with binary classification problems, although multi-class and multi-label adaptations exist (Elisseeff and Weston 2001).

2.1.3 Ensemble Techniques

A common way to improve the predictive performance of classification algorithms is through the use of *ensemble techniques* (Zhou 2012). These techniques induce several base classification models (or classifiers) to predict the classes, and next, the predictions are combined using some procedure. These classifiers must be induced in such a way that they make to some extent different predictions, so that each classifier is better suited to predict different parts of the feature space.

In fact, in order for ensemble techniques to work, two conditions must be satisfied: 1) the base classifiers must be independent from each other and 2) the base classifiers must have better than random predictive performance. In practice, it is difficult to have completely independent classifiers, as the datasets that are used to induce them normally have some overlap, and the base classification algorithms may have a similar classification bias. Also, it is not possible to guarantee better than random performance on the testing set. For these reasons, ensemble algorithms are not always better than their non-ensemble counterparts.

Broadly speaking, there are three independent ways to generate base classification algorithms for ensemble methods: 1) altering the set of instances in the training sets given to the base algorithms; 2) altering the features of the datasets; 3) altering the parameters of the classification algorithm or even using different classification algorithms.

One can wonder why ensemble techniques work at all. To understand this, one can consider the *bias-variance decomposition* of classifier error: it is possible to decompose a classifier error into three sources: *bias*, *variance*, and *dataset errors* (Friedman 1997). Bias is defined as a systematic error that the classifier makes, either due to some broken assumption or because of some non-optimal parameter configuration the algorithm has. This can be seen as the mean classification error the algorithm has if several, independent, classification datasets were sampled from the same distribution and classified by the model. Variance can be seen as how different the predictions of the classifiers can be when several models are induced using datasets samples from the same distribution. Finally, *dataset errors* are errors impossible to avoid due to mislabeled testing instances.

Ensemble techniques reduce classification errors by attenuating the effect of the variance on the final error. That is, by combining the results of several base classifiers, random classification errors that would be made by a stand-alone classifier (without using an ensemble) can be avoided because they are unlikely to occur, at the same time, in the majority of base classifiers - given the previously mentioned assumptions that the classifiers are independent and have better than random predictive performance.

One of the most common ensemble techniques is the *bagging* (or bootstrap aggregation) technique (Breiman 1996), which is the approach of uniformly sampling, with replacement, sub-datasets from the training set. Each sample has the same size of the original dataset and it is used to induce a classifier. The classifiers induced using different samples are usually different because their training datasets will be different: on average, each bootstrap sample will contain approximately 63% of the original dataset. The final prediction is typically retrieved by getting the most voted prediction from the pool of classifiers. Like other ensemble algorithms, bagging is known to improve classification performance of classification algorithms by reducing the overall variance of the base classifier when several decisions are combined.

There are other commonly used ensemble techniques such as *Boosting*, *AdaBoost*, and *Random Forests*, which were not used in this work. Readers interested in different ensemble techniques can refer to (Tan, Steinbach and Kumar 2006; Zhou 2012).

There are many other aspects of the classification problems; however, the previously defined techniques cover most aspects of the classification problems studied in this thesis. Readers interested in other aspects of data mining and classification can refer to the books of Witten and Frank (2000); Tan, Steinbach and Kumar (2006).

2.2 Hierarchical Classification

Among the several types of classification problems discussed earlier, this thesis focuses on hierarchical classification problems.

Hierarchical classification, sometimes called structured classification, is a special type of classification problem in which the classes of the instances have a pre-established hierarchical taxonomy, generally specified as a DAG (Directed Acyclic Graph) or a tree. We shall consider that this taxonomy defines an "Is-a" relationship between classes, that is, if an instance is classified as a class c_a, it is also implicitly classified as all the classes in the set $Anc(c_a)$, where $Anc(c)$ is defined as the set of all ancestors of class c.

Following the notation proposed in (Silla Jr. and Freitas 2011a), hierarchical classification problems may be described by the tuple $\langle \Upsilon, \Psi, \Phi \rangle$, as follows. $\Upsilon \in \{T, DAG\}$ defines the type of structure of the class taxonomy (Tree or DAG, respectively). $\Psi \in \{MPL, SPL\}$ indicates, respectively, whether at least one instance has Multiple Paths of Labels (MPL) or every instance has only a Single Path of Labels (SPL) in the class taxonomy – where a "path of labels" means the set of labels in a path from the root node to the leaf nodes of the class hierarchy. This distinction is analogous to the distinction between single-label and multi-label classification problems in "flat" classification. $\Phi \in \{PD, FD\}$ dictates if at least one instance has Partial Depth (PD) labeling or every instance has Full Depth (FD) labeling. FD labeling requires the most specific class of all instances to be a leaf node, while in PD labeling an instance may have a non-leaf node as its most specific class.

Hierarchical classification algorithms usually use one out of two broad approaches: the local classification approach, and the global classification approach. These approaches are described in detail in the next subsections.

2.2.1 Local Hierarchical Classification (LHC)

The local hierarchical classification approach first uses several traditional (flat) classification models to predict the classes in different small parts of the class taxonomy. These local models are trained by transforming parts of the class taxonomy into several *flat* classification problems by using some class relabeling strategy. Then, in the prediction phase, some procedure is employed to retrieve a coherent, hierarchical classification for an instance.

There are three approaches to train classification models using local classifiers. The first, and most frequent one, is training a binary classifier per node, but the root node, to predict if an instance belongs to each one of the classes in the hierarchy. This approach is exemplified in Figure 2.2. As an example of this approach, Cesa-Bianchi, Gentile and Zaniboni (2006) propose a probabilistic framework to model the instances of a $\langle T, MPL, PD \rangle$ problem and an exact approach to minimise a loss function for the model. They compare their approach to flat and hierarchical variations of SVM classifiers. Their work did not achieve better predictive performance than the baseline techniques (that don't try to optimise the loss function directly), which shows that real-world problems instances are not well behaved and closed-form approaches are not always the best choice to achieve good classification performance.

The work of Valentini (2011), for $\langle T, MPL, PD \rangle$ problems, uses the binary classifier per node approach to predict gene functions using the FunCat class-taxonomy. Another work using this approach is the paper of Xiao et al. (2009) (for $\langle T, SPL, FD \rangle$ problems), in which the authors use local, per-node classifiers to predict emotions in human voice. The work of Barutcuoglu and DeCoro (2006) (for $\langle T, SPL, FD \rangle$ problems) is specially interesting, as the authors use k-NN as base classifiers and a Bayesian network to output a consistent probabilistic classification, by treating each k-NN classifier's prediction as a node in the Bayesian network. The authors even suggest (but did not implement) the use of Gibbs sampling as a way to perform efficient inference, as we did in this work (presented later).

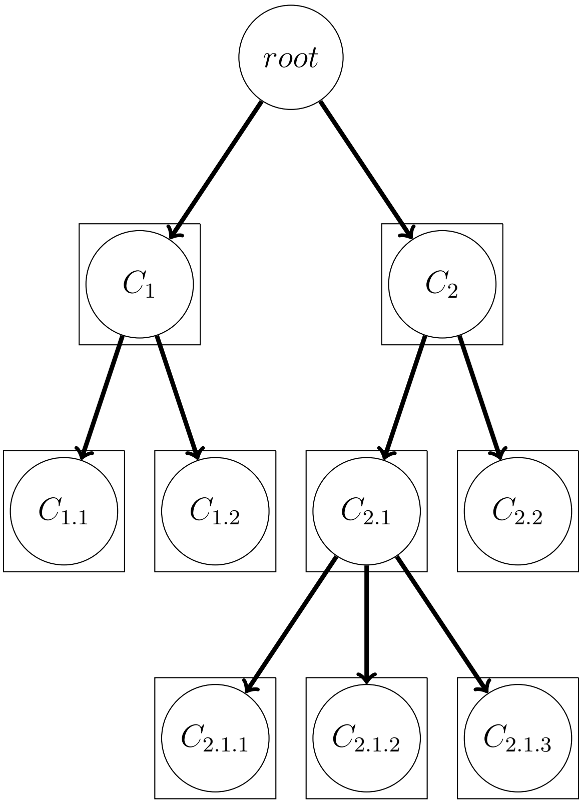


Figure 2.2: Example of per-node local hierarchical classifiers. Circles represent hierarchical class nodes, edges represent 'IS-A' relationships between classes, and each square represents a binary classification model induced to predict membership or non-membership of an instance in each class node (except the root node, since all instances belong to the root node).

The second approach is to train a multi-class or multi-label classifier per parent node (every node except the leaves) that predicts which child class(es) the instance belongs to. A multi-class classifier is used if the problem is SPL, whilst a multi-label classifier is used if the problem is MPL. This approach is exemplified in Figure 2.3. This approach was used in the seminal work of Koller and Sahami (1997), for $\langle T, SPL, PD \rangle$ problems. They used local hierarchical classifiers for the first time, taking advantage of the class hierarchy to perform hierarchical classification. After performing feature selection for each class node, the authors trained several k-dependence Bayesian network models using the local classifier per parent node approach. Another notable implementation of this approach is the work of Alaydie, Reddy and Fotouhi (2012), for $\langle T, SPL, PD \rangle$ problems, in which the authors propose the algorithm HiBLADE, which trains a classifier for each non-leaf class node considering a feature vector, extended with the classes labels of its ancestors. During the testing phase, the authors use a bagging technique that takes in consideration the class taxonomy, using classifiers trained in the children of a node as candidate classifiers for the ensemble. In addition, Hernandez, Sucar and Morales (2013) developed a local classifier per parent node approach for the $\langle T, SPL, FD\rangle$ problem, testing several ways to output a coherent classification.

A third, less common approach, illustrated in Figure 2.4, to train classification models using local classifiers is to use one classifier per level of the hierarchy, as done by Cerri, Barros and de Carvalho (2011). In that work the authors train one Neural Network classifier per level of the hierarchy to solve a $\langle DAG, MPL, PD\rangle$ problem (although the algorithm is also suitable for $\langle T, MPL, PD\rangle$ problems).

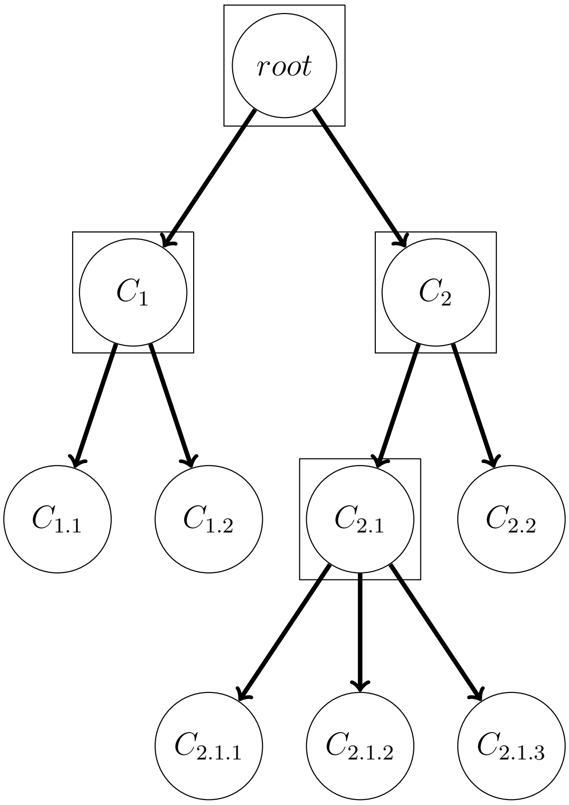


Figure 2.3: Example of per-parent-node local hierarchical classifiers. Circles represent hierarchical class nodes, edges represent 'IS-A' relationships between classes, and each square represents a multi-class or multi-label classification model to predict membership of an instance in zero or more child class nodes.

2.2.2 Global Hierarchical Classification

The definition of global classifiers is more elusive, we consider it to be a non-decomposable classification model that returns the hierarchical classification of an

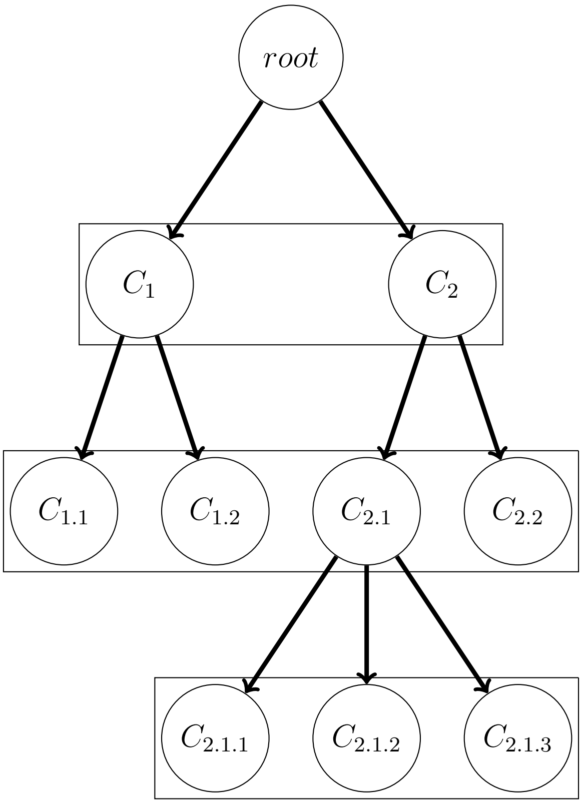


Figure 2.4: Example of per-level local hierarchical classifiers. Circles represent hierarchical class nodes, edges represent 'IS-A' relationships between classes, and each rectangle represents a multi-class classification model induced to predict membership of an instance in zero or more class nodes in a level of the hierarchy.

instance at once, instead of using some strategy to combine separate classifications like in the local classifier approach. The global approach for hierarchical classification is represented in Figure 2.5.

The global classification approach has two advantages over the flat and local ones: 1) reduced model complexity: a single classification model induced to predict all hierarchical classes at once is, usually, much smaller than a collection of local models induced to predict several classes independently; and 2) global models naturally take into account relationships between classes that are not automatically considered in the training phase when using local approaches.

Several global classification algorithms have been proposed to deal with hierarchical classification problems. Wang, Zhou and He (2001) have proposed, in the context of document classification, a global classifier based on association rules. Clare and King (2003) have adapted the decision tree algorithm C4.5 to deal with hierarchical classification problems. Otero, Freitas and Johnson (2010) proposed

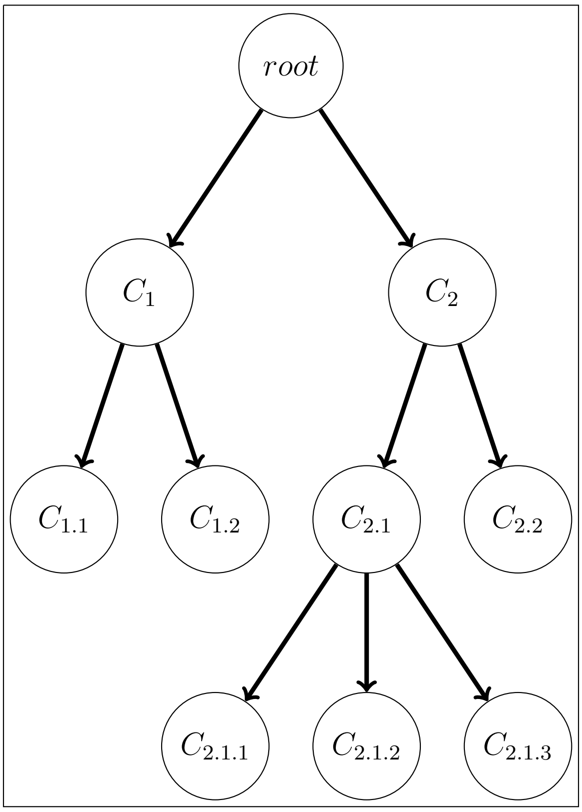


Figure 2.5: Example of a global hierarchical classifier. Circles represent hierarchical class nodes, the edges represent 'IS-A' relationships between terms. The global classification model is represented by a large rectangle, this model is induced to predict all hierarchical classes at once.

an Ant Colony Optimisation (ACO) algorithm to find rules to classify instances in hierarchical classes. Silla Jr. and Freitas (2009) introduced a global version of the Naive Bayes algorithm for $\langle T, SPL, PD\rangle$ hierarchical classification problems. Their algorithm calculates the probabilities of all class nodes, altering it using a usefulness factor that increases the probabilities of nodes near the leaf nodes, motivated by the fact that deeper classes tend to represent more useful knowledge for users.

Another type of global classification algorithm is the adaptation of kernel functions for hierarchical classification problems. After the kernel functions are defined, traditional classification algorithms like SVMs can be used to induce the predictive models (Rousu and Shawe-Taylor 2006; Cai and Hofmann 2007). In (Bi and Kwok 2011), the authors use a kernel projection method to transform the hierarchical classification problem into a set of multi-label classification problems using Principal Component Analysis. A standard algorithm can be used to solve the transformed problem, and next, the solution is re-mapped to the original hierarchical space. The authors concluded that the proposed method is superior to the PCT algorithm, a strong baseline method, which is discussed next.

Overview of Predictive Clustering Tree (PCT) Algorithms

The state of the art in the hierarchical classification of gene and protein functions (considering predictive performance) is the Predictive Clustering Tree Ensemble (PCTEN) (Schietgat et al. 2010) algorithm, for $\langle DAG, MPL, PD\rangle$ problems. This algorithm uses the bagging technique to construct an ensemble of PCT hierarchical classifiers, based on a special type of decision trees (described below). The use of bagging considerably increases the predictive performance of the base algorithm in most cases (although not always, as will be shown in our experiments later). The PCT algorithm was proposed by Blockeel et al. (2002) (for $\langle T, MPL, PD\rangle$ problems) and extended by Vens et al. (2008), to consider DAG-structured hierarchies.

Although the PCT algorithm has worse predictive performance than the PCTEN algorithm in most cases, it has the advantages of being faster and generating a single classification model, which is more easily interpretable than a large set of models in an ensemble of classification models like PCTEN. Vens et al. (2010) expanded the comparison made by Vens et al. (2008) to include more baseline algorithms (like the PCTEN) and datasets. Their conclusion is that the PCT/PCTEN methods outperform the previous state-of-the-art hierarchical classification algorithms (Clare and King 2003; Barutcuoglu, Schapire and Troyanskaya 2006; Vens et al. 2008) with respect to several criteria (interpretability and model size; predictive performance; and induction time).

We use both the PCT and PCTEN classifiers as baselines in the predictive performance analysis of our proposed algorithm. For this reason, next we explain in more detail the induction and classification procedures of both algorithms.

Description of the Predictive Clustering Tree (PCT) Algorithm

The PCT classification algorithm was originally developed to work with flat multi-label classification problems (Blockeel, De Raedt and Ramon 1998), and later extended to work with tree-structured hierarchical classification problems (Blockeel et al. 2002) and DAG-structured hierarchical classification problems (Vens et al. 2008). In this section we shall present the version of the PCT classification algorithm that supports DAGs, which is the most generic and most popular version of the algorithm.

At a high level of abstraction, the PCT algorithm works like most decision tree inducing algorithms (e.g. C4.5 (Quinlan 1993)): when processing continuous attributes, it finds a condition involving an attribute and a constant (e.g. some attribute greater than a constant value) that splits the data into two disjoint groups (or clusters) maximising the dissimilarly of the class labels in each one of the two clusters. If the chosen attribute is discrete (or nominal), the algorithm uses all possible attribute values to split the dataset typically creating a new group (or cluster) of instances for each of the values of the attribute. If the attribute is continuous, the attribute values are ordered and every possible interval between two consecutive values is tested. Next, the algorithm recurses into each one of the clusters until a given stopping criterion is met. When every cluster is fully explored, the classification algorithm returns the fully parameterised decision tree model. The pseudo-code (based on the code presented in (Vens et al. 2008)) formalising this explanation is presented in Algorithm 2.1.

Algorithm 2.1 Pseudo-code of the PCT classification algorithm.

```
1: procedure PCT($D$)
2:     $(t^*, P^*)$ = BestTest($D$)
3:     if $t^* \neq none$ then
4:         for each $D_k \in P^*$ do
5:             $tree_k$ = PCT($D_k$) /* Recursive call to PCT /*
6:         end for
7:         return node($t^*$, $\bigcup_k \{tree_k\}$)
8:     else
9:         return Leaf(ClassProbVect($D$))
10:    end if
11: end procedure
```

This algorithm receives as a parameter the set D, that represents the training instances, and returns a decision tree. This set comprises tuples on the form (x_j, P_j), where x_j is the predictive feature vector of the j-th instance and P_j is the binary class label vector of the j-th instance. This vector encodes the class-membership of the instances for each one of the classes. This vector has the value 1 in the i-th position if the instance is annotated with the i-th class, and 0 otherwise.

At line 2 the *BestTest* function chooses the best split (partition) (P^*) and the corresponding test condition ($t*$) that induces the partition P^*, iterating over all possible splits and using the following measure of dissimilarity between the clusters created by a split in order to choose the best split:

$$Dis(D) = Var(D) - \sum_{D_k \in P} \frac{|D_k|}{|D|} Var(D_k) \tag{2.1}$$

Where D is the dataset in the current tree node and P is a set containing the sub-datasets (clusters) defined by the split under consideration. The higher the value of the measure $Dis(D)$, the better the split. The variance of the training instances Var(D) (and Var(D_k)) is defined in terms of the Euclidean distance between each binary class label vector and the mean class label vector of the partition. Refer to (Vens et al. 2008) for a detailed explanation on how this measure is calculated, including the adaptation for DAG-structured hierarchies.

In addition to choosing the split with highest dissimilarity between the just created clusters using Equation 2.1, the PCT algorithm also checks if the splits are statistically significant. In order to do that, (Blockeel et al. 2002) suggest using an F-test, and only accepting splits with an F statistic greater than s (a user defined parameter). Note that the value of s implicitly regulates the size of the decision tree: smaller values will generate decision trees with equal or greater sizes than bigger values. The F-test statistic is calculated as follows:

$$F = \frac{SS/(n-1)}{(SS_L + SS_R)/(n-2)} \tag{2.2}$$

Where SS is the sum of squared differences between the mean class label vector and the class label vector of the instances in D, and SS_L and SS_R have the same meaning, but for the two (left and right) clusters formed by the split $t*$.

If the F-test fails, the value of t^* is *none*, and line 9 is executed, returning the class probability vector that represents the class labels in D. This class probability vector will function as the prediction of the PCT model (a leaf node of the decision tree). During the testing phase, if the testing instance satisfies all conditions leading to this leaf, this class probability vector will work as the probabilistic predictions of that instance. In practice, each element of the class probability vector is calculated by averaging the binary class label vectors of the instances in D.

If the F-test does not fail, t^* will contain the best test that was found by the BestTest function. Because t^* is not equal to *none*, line 4 is executed, and the algorithm iterates over the two partitions, recursing in each one at line 5. After the *for* loop is executed, at line 7, a new decision tree node is returned, containing the subtree generated by the recursive call at line 5 and the associated test t^*.

A fully parameterised decision tree will be returned by the first call to PCT. To use the model, the user presents an instance to the decision tree, which will return the class probability vector (the mean binary class label vector) associated with the training instances of the cluster where the testing instances falls. The predictive class probability vector associated with the testing instance is the probability that the instance belongs to each one of the hierarchical classes. Because of the way the clusters are constructed, this probability vector is guaranteed to be consistent, that is, the predicted class probability of a parent class will be always greater or equal than the probability of its children.

One of the main advantages of using the PCT classification algorithm is that the generated model allows for a direct interpretation of the learnt associations between the predicted class labels and the predictive features. This is particularly useful in domains such as bioinformatics, where sometimes new, potentially useful, knowledge may be extracted from the classification model.

Description of the PCTEN Algorithm

The PCTEN algorithm induces several PCT decision trees using the bagging ensemble technique (Breiman 1996). This technique works by creating several bootstrap replicates of the training dataset, sampling, with replacement, instances from the original training dataset.

In the testing phase, the predictions of each decision tree are retrieved and the mean class probability vector across the predictions is returned.

The PCTEN has been shown to be in general clearly superior to the PCT algorithm in terms of predictive performance (Schietgat et al. 2010); which is expected, since ensemble methods are known to improve the predictive performance of several classification algorithms (Opitz and Maclin 1999; Hamza and Larocque 2005). The improvement in performance, however, arguably comes with the price of reduced interpretability (Freitas 2013). In other words, the PCT classifier is easier to interpret than its ensemble version, which justifies testing both PCT and PCTEN classifiers.

2.3 Probabilistic Graphical Models (PGM) for Classification

In many domains it is useful, if not required, to interpret the induced classification models. Because PGM classification algorithms are capable, at least in principle, of generating interpretable models (rather than "black-box models"), they have been extensively used in classification tasks with interpretability requirements.

In the next sections we shall focus specifically on two types of PGMs for classification: the commonly used Bayesian Network (BN) classifiers and their variants; and the less used Dependence Networks (DN) classifiers, which are the focus of this thesis.

2.3.1 An Overview of Bayesian Network Classifiers

Bayesian Networks (BNs) were one of the first PGMs widely used for automated inference. The book of Pearl (1988) if often cited as the first comprehensive review of the several techniques necessary to build and query BNs. BNs are defined by two types of components: 1) their *structure* and 2) the *Conditional Probabilities Distributions* (CPDs) of each node.

Structurally, BNs are defined using Directed Acyclic Graphs (DAGs), where each node represents a random variable and each edge represents a (potentially) causal relationship between two random variables. That is, if there is an edge pointing from one random variable to another, knowing the value of the former variable affects our belief about the value of the latter. Each node (random variable) of a BN has a CPD that models the probability distribution of that node given the values of its parent nodes (random variables that point to the node).

An example of a BN is shown in Figure 2.6. This figure shows, for instance, that the student grade is directly influenced by the test difficulty and his or her intelligence. Note that, although the student grade cannot influence the test difficulty, the former variable can still be useful for predicting the latter. However, this predictive relationship cannot be represented in a Bayesian network by an edge pointing from the student grade to the test difficulty, because this would create a cycle in the network. In addition, to predict the test difficulty, one also needs to know the student intelligence, and this important relationship is not explicitly shown in the BN.

BNs may be directly applied to classification; however, it is often reported that

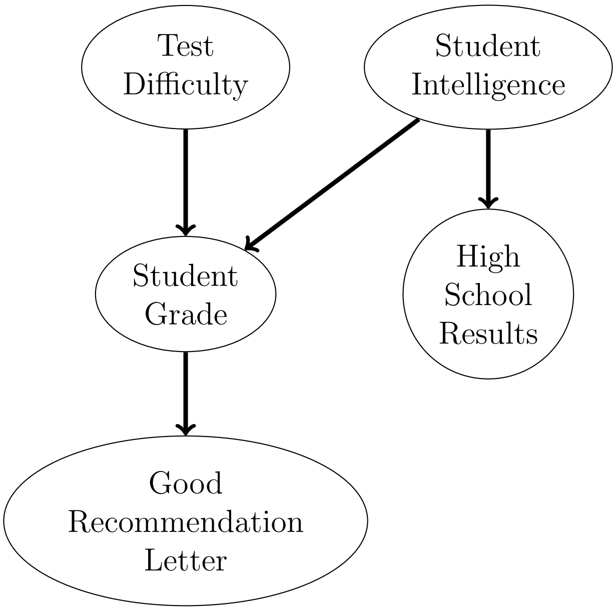


Figure 2.6: A hypothetical Bayesian Network (BN) modeling the random variables that affect the grade of a student in a test and the probability of getting a good recommendation letter. This figure is adapted from an example in (Koller and Friedman 2009).

modeling relationships among predictive variables that do not directly affect the prediction of the class often results in over-fitted classification models (Cheng and Greiner 2001).

For this reason, variants of BNs tailored for the classification task, called BN classifiers, have reported better predictive performance. The idea behind these models is to adopt some simplifying assumption to reduce the complexity of the models, thus minimising over-fitting. The Naive Bayes (NB) algorithm was probably the first algorithm used for this end (Duda and Hart 1973). Other widespread alternatives include the Bayesian Augmented Naive Bayes (BAN) (Friedman, Geiger and Goldszmidt 1997) and Tree Augmented Naive Bayes (TAN) (Friedman, Geiger and Goldszmidt 1997) algorithms.

The main differences between NB, TAN and BAN are as follows. The NB classification algorithm is the simplest one, making the strongest assumptions among the three. The models generated by the NB classification algorithm assume that the features are independent from each other given the class, that is, it assumes that there are no dependencies between the random variables that represent the features. This is a very strong assumption that greatly reduces the number of possible models that the NB classification algorithm can generate, which often leads to worse predictive performance compared to more flexible models. However, the NB has the advantage of low computational complexity and implementation simplicity.

The formula used to calculate the unnormalised class probability values of an instance given its features values is:

$$P(C_i|X) \propto P(C_i) \prod_{j=1}^{n} P(X_j|C_i) \tag{2.3}$$

Where X is the predictive feature vector of the testing instance, C_i is the i-th class label, $P(C_i)$ is the prior probability of class label C_i, and $P(X_j|C_i)$ the probability of the j-th feature taking the value X_j given that the instance is annotated with the class label C_i. This probability is normally estimated using simple counting after discretisation (if the feature values are continuous).

The standard TAN classification algorithm considers that, unlike the NB classification algorithm, there are dependencies between the features of the classification problem. Specifically, the TAN model will consider that the connections among feature (nodes) form a tree involving *all* features. The connections of this tree can be found using a greedy algorithm that connects the two nodes (using an undirected edge) with largest mutual information score given the class, as long as this new connection does not induce a cycle among the feature nodes. After this step, a node is selected at random to be the root of the feature tree, defining the direction of the previously undirected edges connecting the features. Models generated by the TAN classification algorithm have, generally, better predictive performance than models generated using the NB classification algorithm. However, this improved predictive performance comes with a higher computational cost. Also, it has been argued that forcing the connection of every predictive feature may give rise to unnecessary connections in the graphical model, which can harm performance (Keogh and Pazzani 1999).

The formula used to calculate the unnormalised class membership probability for TAN, similar to the one used for NB, is as follows:

$$P(C_i|X) \propto P(C_i) \prod_{j=1}^{n} P(X_j|Pa(X_j), C_i) \tag{2.4}$$

Where $Pa(X_j)$ represents the set of feature values for the parents of the j-th feature, and the other terms are as defined for Equation 2.3. Again, these probabilities are estimated by simple counting.

Finally, in models generated by the BAN classification algorithm (as the models generated by NB and TAN) there is an edge from the node representing the class variable to every node that represents a feature. But unlike the models generated by the TAN algorithm, the BAN classification algorithm does not restrict the connections of the features to form a tree, allowing for the formation of DAGs. This increases the search space of the BAN algorithm in relation to the TAN, but has the potential for finding models that fit the data better. Normally, BAN algorithms have a parameter (k) that controls the maximum number of parents a feature can have, to avoid over-fitting.

The formula used to calculate the unnormalised class membership probability of BANs is the same as for TANs, shown in Equation 2.4.

Figure 2.7 shows one example for each of the three variations of BN-inspired classification algorithm that were previously explained (NB, TAN and BAN). Note that that the model generated by the NB algorithm does not contain edges among the predictive features, while the edges among predictive features in the classification model induced using the TAN algorithm induce a tree. Finally, the edges among the predictive features in the model induced by the BAN algorithm form a DAG.

Regarding hierarchical classification, Silla Jr. and Freitas (2009) have introduced the global hierarchical version of the NB classifier and tested their approach in several datasets for $\langle T, SPL, FD \rangle$ problems, reporting a better performance than using local Naive Bayes classifiers. Barutcuoglu and DeCoro (2006) have used BNs to integrate the decision of several local classifiers, correcting possibly inconsistent classifications; this approach works for $\langle DAG, MPL, PD \rangle$ problems. In Barutcuoglu, Schapire and Troyanskaya (2006), the authors used the same technique to predict the function of genes using a GO taxonomy. Campos, Fern and Huete (2006) use BNs to predict the categories of web documents considering a DAG-structured class hierarchy.

2.3.2 Dependence Networks

Dependence Networks (DN) are a relatively new type of PGM first described by Heckerman et al. (2001). Like BNs, each node of a DN must encode a probability distribution conditioned on the values of is parents. However, DNs are a more generic alternative than BNs since they allow for cycles in their graphical representation.

Like a BN, a DN comprises two parts: 1) the structure of the DN and 2) the

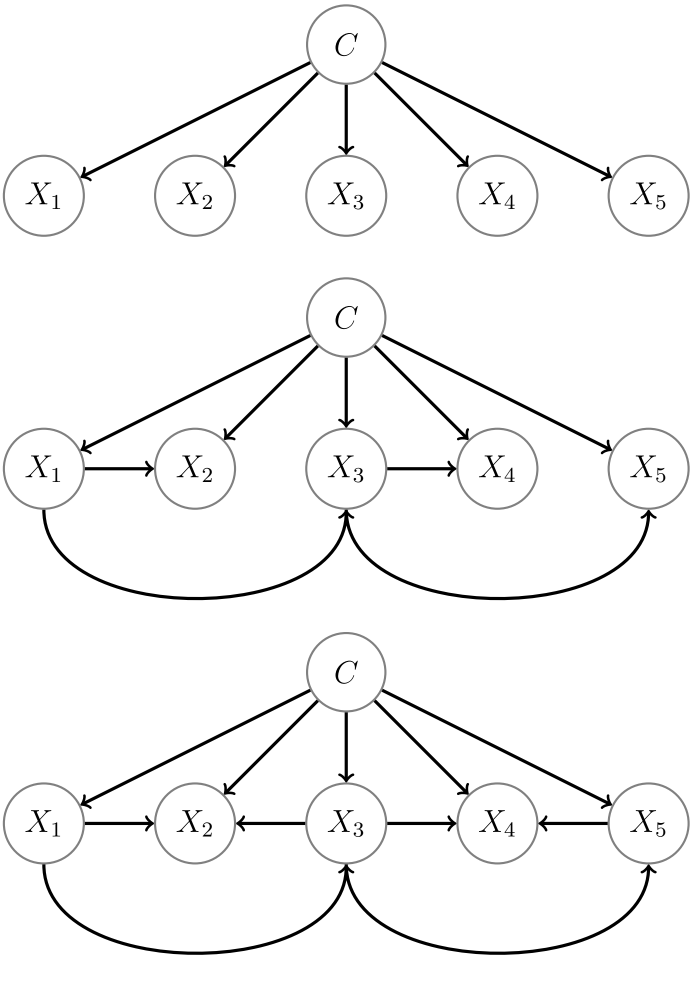


Figure 2.7: Examples of classification models generated using the NB, TAN, and BAN classification algorithms, from top to bottom. The node C represents the class variable, and the nodes from X_1 to X_5 five predictive features. Edges represent dependencies between variables. Note that in the model generated by the NB classifier there are no links between the nodes of the predictive features, while the links in the nodes of the predictive features in the models generated by the TAN and BAN form a tree and a DAG, respectively.

Conditional Probability Distribution (CPD) of each random variable. In DNs the graphical structure is not restricted to DAGs like in BNs, so it can contain cycles. Also, the meaning of the edges is different: in BNs the edges represent potentially causal relationships among variables, while in DNs they represent dependencies among variables. Such dependencies are useful for prediction purposes, but there is no intention of trying to represent causality. Like in BNs, the CPDs can be modelled using probability distribution tables or other types of function.

An example of a DN is shown in Figure 2.8, to be contrasted with its counterpart in Figure 2.6 (for BNs). Note that in the DN there is an explicit edge between the test difficulty and the student intelligence random variables. In addition, the relationship between test difficulty and student grade is bi-directional (and therefore cyclic), i.e., each of them can be used to predict the other. This is the case because to determine the test difficulty, ones needs to know both the student grade and the student intelligence, since knowing only either information is not enough to determine the test difficulty. This additional flexibility, including the ability to represent cyclic relationships, helps users of DNs understand the underlying correlations among random variables more easily.

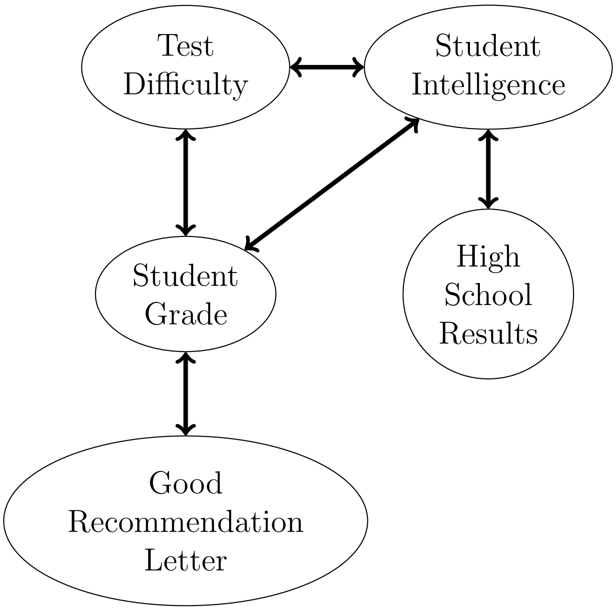


Figure 2.8: A hypothetical Dependence Network (DN) modeling the same random variables in the BN presented in Figure 2.6.

The main difference between DNs and BNs is that in DNs the neighbours of a random variable encode the Markov blanket of a node (Koller and Friedman 2009), or in other words, the set of random variables that make the random variable represented by that node independent from all other random variables. A BN can be transformed into a DN by switching every directed edge to a non-directed edge and, for each node, including non-directed edges from that node to the parents of the children of that node.

In fact, consistent DNs are equivalent to another type of PGM, namely, Markov Networks (Koller and Friedman 2009), sharing the same prohibitive time complexity for inference and parametrisation (inference and parametrisation in consistent DNs is a NP-hard problem).

Parametrisation of DNs involves two steps: *structure learning* and *conditional probability estimation* of the random variables. Like the case of BNs and Markov networks, learning the optimal structure of DNs is a very computationally expensive problem. The number of possible network structures increases exponentially with the number of random variables whose probability distribution ones wishes to estimate. In fact, as previously stated, learning the optimal structures of the three aforementioned types of PGMs is NP-hard (Koller and Friedman 2009). Therefore, for moderately large problems, learning the structure optimally is not feasible. To overcome this computational limitation, Heckerman et al. (2001) proposes the use of general Dependence Networks (which we call simply DNs from now on), which are bounded approximations for consistent DNs that have efficient parametrisation and inference phases.

One possible strategy to avoid prohibitive running times is to use specialist (domain-specific) knowledge to model the correlations between random variables. This was the approach employed in the first version of our Hierarchical Dependence Network (HDN) algorithm, to be presented in detail in Chapter 5. Using specialist knowledge has the advantage of avoiding computationally expensive structure learning algorithms but has the vulnerability of relying on potentially incorrect and/or incomplete specialist knowledge.

Another strategy is to use heuristics to find reasonably good structures using a data-driven approach. As an example of a heuristic approach, in Heckerman et al. (2001) the authors propose using feature selection algorithms to find the structure of the DN. For each class C_i they build a classifier to predict C_i using both the features and other classes, which are called predictive classes. Hence, they extend the feature vector of the instances with all predictive classes and apply a feature selection algorithm to get the predictive features and predictive classes that influence prediction of class C_i (the Markov blanket of C_i). The final structure of the DN will have an edge pointing from each class variable in the Markov blanket of C_i to C_i, for every class C_i. Approaches that use feature selection have the advantage of relying on well-studied and easily available feature selection algorithms. However, the most computationally efficient feature selection algorithms tend to consider only bivariate tests of independence, measuring the association between the class variable and just one feature at a time, which fails to detect more complex relationships among random variables.

In (Neville and Jensen 2007; Tian et al. 2006) the authors use classification algorithms that have some type of implicit (or embedded) feature selection (like decision trees) to model each class and use the implicit feature selection process to determine the structure of the DN. That is, when inducing a classification model to predict class C_i, the predictive feature vector is extended to contain the other classes, used as predictive classes. There will be an edge from some other class C_j (in the extended feature vector of C_i) to C_i if the model uses C_j to predict C_i. Relying on classification algorithms to detect the relations between random variables has the advantage of not depending on a feature selection algorithm used in a pre-processing phase, which may select a feature set that has lower predictive performance. On the other hand, one should be careful to avoid classifier over-fitting when relying on a classification algorithm to select the predictive features, specially when the number of classes is large.

Another approach, presented in (Gámez et al. 2008; Gámez, Mateo and Puerta 2006) is to use tests of statistical independence, such as the G^2 test or χ^2, to directly find the Markov blankets of the classes and thus, determine the structure of the DN. Note that these tests of independence are normally bivariate, only checking if pairs of random variables are independent, not considering if one random variable is independent of another given a third random variable, as considering such dependencies would be too computationally expensive. On the other hand, statistical tests have the advantage of a solid mathematical basis.

The work of (Guo and Gu 2011) ignores the problem of finding the structure of a DN and uses a fully connected DN, delegating the task of dependence estimation to the set of classifiers, each of which predicts the value of a different class variable. Note that this approach is more sensitive to classifier over-fitting, as it requires a classification algorithm that is able to “ignore” uninformative features and uniformative predictive classes. On the other hand, this approach avoids the potentially computationally expensive step of finding the structure of DN directly.

Once the structure of a DN is defined, we need to estimate the *conditional probabilities* of the classes. The usual way to achieve this is to use a classification model to estimate the conditional probabilities. For instance, in (Heckerman et al. 2001; Gámez, Mateo and Puerta 2008) the authors use decision trees to estimate the conditional probabilities; in (Gámez, Mateo and Puerta 2006) the authors use the Naive Bayes classifier; and in (Guo and Gu 2011) the authors use a logistic regression classification algorithm.

DNs have been used in several non-traditional classification problems: Toutanova and Klein (2003) used DNs for Part-of-Speech tagging. The authors claimed that the ability of the DN for representing cyclic relationships among variables in the hierarchy contributed to the good performance of their algorithm in relation to the state of the art. The work of Tian et al. (2006) explores DNs in the field of link-based classification of documents, which can be defined as the classification of linked objects under the assumption that the classes of linked documents affect each other. This work reported that the use of DNs increased predictive performance compared to baseline approaches. Neville and Jensen (2007) have used DNs in the task of collective classification, which may be defined as the collective prediction of correlated instances into a set of classes.

Regarding more traditional classification problems, Gámez, Mateo and Puerta (2006) have used a DN in the task of multi-class classification. In this work the authors use a DN with a Naive Bayes classifier to learn the conditional class probabilities associated with each node and a χ^2 statistical test to estimate the Markov blanket of each node. Later on, the authors enhanced their approach by using a specialised DN to predict each class, additionally they proposed a mechanism to improve the time complexity and confidence of the determination of the Markov blanket of each class label (Gámez et al. 2008). The authors also reported that the flexibility of DNs was responsible for improving the performance of their algorithm in relation to baseline approaches.

More recently, Guo and Gu (2011) proposed a method for multi-label classification using DNs. In their case Gibbs sampling (briefly described below) was necessary to model relations between classes. The authors reported superior performance compared to several baseline multi-label classifiers.

Gibbs sampling is a Metropolis-Hastings style algorithm that is used to make inference in DNs and other PGMs. Gibbs sampling works as follows. First, it randomly assigns values to the unknown random variables present in the network. Next, in the main loop, the algorithm proceeds to execute two phases: the *burn-in phase* and the *sampling phase*. In the burn-in phase, in each iteration, the algorithm considers all random variables, one at a time, and it re-draws the value of each variable using the current state of the Markov blanket of that random variable as the ground-truth. In other words, the burn-in phase consists of running the main loop of the Gibbs algorithm for a pre-defined number of iterations without considering the values of the variables for the inference of the DN. This is necessary for the probability estimates to converge to a stationary distribution.

After the burn-in phase, the *sampling phase* starts. In this phase the states of the random variables are sampled. After a given number of *sampling iterations*, the algorithm averages over the sampled random variable values and returns this average as the network's probabilistic inference (Heckerman et al. 2001).

It is important to note that although Gamez et al. use DNs for classification, (Gámez, Mateo and Puerta 2006) do not use Gibbs sampling for inference. This is not necessary since they are in the multi-class scenario and do not need to consider relations between multiple classes occurring at the same time.

As far as we know there is no work exploring Dependence Networks in the Hierarchical classification setting, as proposed in Chapter 5 of this thesis.

2.4 Related Work Exploring Class Label Dependencies in Hierarchical Classification

There are several works dealing with hierarchical classification that explore dependencies among class labels. In the next sub-sections we deal with two types of approach that use class dependencies: *prediction consistence maintenance approaches* and *prediction propagation approaches.*

Prediction consistence maintenance approaches employ some strategy to "fix" inconsistent predictions (usually made by local classifiers) across class nodes. The meaning of 'inconsistent predictions' varies depending on whether the hierarchical classification model predicts the class label membership probabilities of the instances or it predicts the actual class label that the instances are annotated with (a 'crisp' classification).

For probabilistic classification outputs, the predictions are inconsistent if the class-label probability of a given class in the hierarchy is greater than the probability of any of its ancestors. The predictions are inconsistent for crisp classification if the classifier annotates an instance with some class label in the hierarchy, but does not annotate that instance with all the ancestors of that class label.

Prediction propagation approaches, on the other hand, employ some strategy to use the predictions of some class nodes during the prediction of other nodes, normally using the class predictions as extended features.

The main difference between these two approaches is that *prediction consistence maintenance* algorithms usually adjust the predictions of the hierarchical classification algorithm after the classifications are made using the implicit class relationships defined by the class hierarchy. So, although there is some information flowing between class labels, this information is not taken in consideration during the prediction phase of the base classifiers. Techniques based on the *prediction propagation approach* on the other hand, use the predictions of other base classifiers during their prediction phase. We apply both techniques in our DNs proposed in Chapter 5.

2.4.1 Exploring Class Label Dependencies for Prediction Consistence Maintenance

In (Barutcuoglu and DeCoro 2006) the authors use the Bayesian network framework to fix the inconsistent predictions made by k-NN classification algorithms induced to predict each class label separately. Each class prediction (as output by k-NN) is represented as a node in the Bayesian network, and the network's edges represent dependencies among the predictions of the base k-NN classifiers. The proposed method outputs the *most probable* consistent predictions according to the Bayesian network. They conclude that their approach always improves the classification performance of their base k-NN classification algorithms.

In (Bi and Kwok 2011) the authors model the prediction consistence maintenance problem as an optimisation problem that can be solved optimally and efficiently using the algorithm "Condensing Sort and Select". The authors shown that their approach achieves better predictive performance when compared to the PCT hierarchical classification algorithm, which will be described in detail in Chapter 5.

In (Cesa-Bianchi, Gentile and Zaniboni 2006) the authors propose a probabilistic framework to model hierarchical classification problems and propose an approach to minimise a loss function for the model, given the predictions of local classifiers. Their approach also fixes inconsistent predictions. They compare their approach to flat SVMs and hierarchical SVMs. Their work did not achieve better predictive performance than the baseline techniques (which don't try to optimise the loss function directly).

The works of (Valentini 2009; Valentini and Cesa-Bianchi 2008; Valentini and

Re 2009) propose variations of a simple algorithm to ‘fix’ inconsistent predictions. Their algorithm scans the class hierarchy from the root node to a leaf node and uses a simple heuristic to adjust the predictions when necessary. They conclude that their approach consistently outperforms an algorithm with non-adjusted predictions.

In (Obozinski, Lanckriet and Grant 2008) the authors test 11 prediction consistence maintenance methods for hierarchical classification. The main conclusions are that, as expected, prediction consistence maintenance tends to improve predictive performance, but not always. In addition, in general, Bayesian approaches do not perform so well, compared to other approaches, including simple heuristics.

2.4.2 Exploring Class Label Dependencies for Prediction Propagation

In (Merschmann and Freitas 2013) the authors propose an Extended Local Hierarchical Naive Bayes (ELHNB) classifier to predict protein and gene functions. Their approach considers relationships between child/parent classes when constructing the classification model. Their work, however, is limited to $\langle T, SPL, FD \rangle$ problems. This method will be further discussed and extended to DAG problems in Chapter 4, and then used in the experiments reported in Chapter 5.

In (Ramírez-Corona, Sucar and Morales 2016), the authors propose the Chained Path Evaluation method for hierarchical classification, which uses the prediction of the parent classes as features, a pruning mechanism to perform non-mandatory leaf node predictions and an algorithm to retrieve the most probable path from the root to a node in the class hierarchy. The authors tested the algorithm in several hierarchical classification datasets and it was statistically equivalent to the baseline algorithms in most tests.

In (Mayne and Perry 2009) the authors use a Naive Bayes algorithm, following the local hierarchical classification approach, to predict hierarchically organised document classes and use the probabilities of the parent classes as features for predicting the child classes.

In (Cerri et al. 2016; Cerri, Barros and de Carvalho 2014), the authors develop a hierarchical Neural Network algorithm, using one neural network per class level and using the output of one level as the input of the next. They report that their algorithm is better than several strong baselines and the previous version of their own algorithm.

2.5 Summary

This chapter first provided a broad overview on the data mining background required to grasp the contents of the rest of this thesis. This chapter started by first reviewing basic concepts of the classification task at Section 2.1, defining the mathematical notation that will be used in the remainder of this thesis; the most common types of classification problems (binary, multi-class and multi-label); and other basic data mining concepts such as feature extraction/selection and the predictive performance evaluation of classification algorithms.

After these concepts are properly defined, we went about explaining how traditional decision tree classification algorithms work in Section 2.1.1. This is essential to the rest of this thesis because we have used several variations of hierarchical decision tree classification algorithms in our experimental study. We explained the basic idea behind the decision tree induction process and how the model can be interpreted by the users.

In Section 2.1.2 we explored basic concepts of 'flat' Support Vector Machines (SVMs), including the intuition behind the training and testing phases and their overall runtime complexities. SVMs are extensively used throughout this thesis to model probabilistic class distributions in our hierarchical classifiers.

To conclude the presentation of traditional data mining techniques used in this thesis, we presented in Section 2.1.3 the concept of classifier ensemble. This is a common technique to improve predictive performance that reduces the variance of single classification models by using several diverse models and combining their predictions. These models are generated varying some aspect of the training procedure. These techniques will be explored later in the context of hierarchical classification.

After these basic 'flat' classification concepts were laid out, we started exploring, in Section 2.2, more advanced aspects of data mining, starting with 'hierarchical classification', defining several aspects of this type of classification problem. We have listed the most traditional approaches to deal with hierarchical classification, categorising the hierarchical classification algorithms and problems in a systematic way. Next we explained the workings of two state-of-the-art hierarchical classification algorithms (PCT and PCTEN) that will be extensively used as strong baselines in this thesis.

In Section 2.3 we explored concepts related to Probabilistic Graphical Models (PGMs) for 'flat' classification, focusing on Bayesian Network models and the (less used) Dependency Network models. This section explained how these models are constructed (how the structure and the parameters of the models are determined) and what are the limitations and strengths of these two types of PGMs.

Lastly, in Section 2.4 we reviewed some works that are related to exploring class label dependencies to improve the performance of both 'flat' and hierarchical classification algorithms. We separate these works into two broad types: works exploring dependencies for maintaining the consistency of predictions across hierarchical classes and works using dependencies to use the predictions made in one class to improve the predictions of some other class. The later type of dependency exploration will be used in some hierarchical classification algorithms proposed by us later on.

Chapter 3

Background on Bioinformatics and Ageing

3.1 Bioinformatics

The term bioinformatics was coined by Paulien Hogeweg in the early 1970's, meaning "the study of informatics processes in biotic systems" (Hogeweg 2011). In the context of genetics, this definition evolved after the "data deluge" of genomic information that started in the late 1980's to mean "the development and use of computational methods for data management and data analysis of sequence data, protein structure determination, homology-based function prediction, and phylogeny"(Hogeweg 2011). In a broader sense, "bioinformatics" took the meaning of the "research, development, or application of computational tools and approaches for expanding the use of biological, medical, behavioral or health data, including those to acquire, store, organize, archive, analyze, or visualize such data (Huerta et al. 2000)."

In this literature review we shall focus on the techniques to annotate genes /proteins and the most used open databases of genomic information used for this end. The early techniques to annotate unknown-function genes/proteins consisted in using homology-based transfer sequence-alignment algorithms such as BLAST (Altschul et al. 1997). The idea was, given an unannotated gene/protein sequence, to find the most similar annotated sequence and transfer its annotations to the unannotated sequence. One of the problems with this approach is that similar sequences may have very different functions and the same functions may be performed by genes/proteins with different sequences (Friedberg 2006; Cai 2003).

Although still popular, these techniques are being relatively less used at present, given space to more sophisticated algorithms that use other features rather than only the sequence information (Friedberg 2006) and rely on more complex techniques to use machine learning to classify large-scale biological data (Bacardit and Llorà 2013; Swan et al. 2013). This fact is evident when analysing the results of the latest Critical Assessment of Functional Annotation (CAFA) competition[1] (Jiang et al. 2016), in which the standard sequence alignment algorithm was consistently outperformed by more sophisticated machine learning techniques that use several sources of information.

For instance, one common approach is to use the existence of certain *motifs* in the protein sequence to infer the function of the protein (Friedberg 2006). Motifs are patterns that appear in different proteins and are somehow related to their function. There are many public motifs databases, for instance Pfam (Finn et al. 2008), Prosite (Sigrist et al. 2013), PRINTS (Attwood et al. 2003) and Interpro (Hunter et al. 2012). Motifs are normally represented as *(flexible) patterns* (where one implicitly defines a set of amino acid sequences that match the motif) or as *Hidden Markov Models (or profiles)* (where one defines a model that returns the probability that a given amino acid sequence matches the motif).

One important recent development in the field of bioinformatics is the development of controlled vocabularies and a structured, well-defined, organisation of terms defining gene/protein functions. The Gene Ontology (GO) (Harris et al. 2004) and the MIPS Functional Category (FunCat) (Ruepp et al. 2004) are solutions to this problem. They define standardised class hierarchies and terms so that functions of genes and proteins may be annotated with reasonable precision.

3.1.1 The Gene Ontology

The Gene Ontology (GO) project (The Uniprot Consortium 2007) aims to provide a comprehensive and freely available resource to annotate the functions of genes and its products. The GO comprises a structured and controlled vocabulary to classify several aspects of gene functions. The ontology is continuously updated and curated to reflect the current knowledge of biologists about the genes of several organisms.

The GO hierarchy is organised as a Direct Acyclic Graph (DAG), with edges representing "Is-A" and "Part-Of" relationships between terms. The nature of

[1] `http://biofunctionprediction.org/cafa/`

these relationships implies that if a gene is annotated with a term, by definition, all ancestors of that term are implicitly assigned to the gene as well. This is known as the “true path rule” and is one of the factors that hinder the use of independent classifiers for hierarchical classification: it is not trivial to transform several classifications that violate the “true path rule” into a consistent one.

Figure 3.1 shows a small part of the GO containing all ancestors of the GO term GO:0006139 (nucleobase-containing compound), with edges representing ‘IS-A’ relationships between terms. Note that, traditionally, the edges of DAGs represent super-type relationships, that is, there is an edge from one node to another node if the first node is a general category of the second node. However, because the GO considers ’IS-A’ relationships, the directions of the edges are inverted, pointing from child node to parent node, that is, there is an edge from one node to another node if the first node is a sub-category of the second node.

Generally speaking, the GO has 3 independent types of annotations: 1) Molecular Function (about 10,000 terms) – annotations that describe activities of the products of a gene at the molecular level, 2) Biological Process (about 25,000 terms) – annotations that describe biological goals accomplished by the gene, and 3) Cellular Component (about 3,000 terms) – annotations that describe the locations where gene products are active at the cellular level.

The use of GO terms to annotate gene and protein functions has become a standard in the field of protein function prediction, being broadly used in many works in the area of hierarchical function prediction of genes and proteins.

3.1.2 The FunCat Class Hierarchy

The FunCat (MIPS Functional Catalogue) (Ruepp et al. 2004) may be described as an organism-independent hierarchical classification scheme for the description of gene and protein functions. Unlike the GO, the FunCat hierarchy is organised as a tree, not a DAG. Trees are arguably easier to manage and, therefore, are the representation of choice in some domains. The FunCat hierarchy provides a coarser level of information than the GO annotation, therefore, being easier to maintain and interpret.

The main categories of the FunCat catalogue are: metabolism, information pathways, transport, perception and response to stimuli, developmental processes, localisation, and experimentally uncharacterised proteins. The depth of the FunCat hierarchy is much shallower than GO: the largest path from the root to a leaf

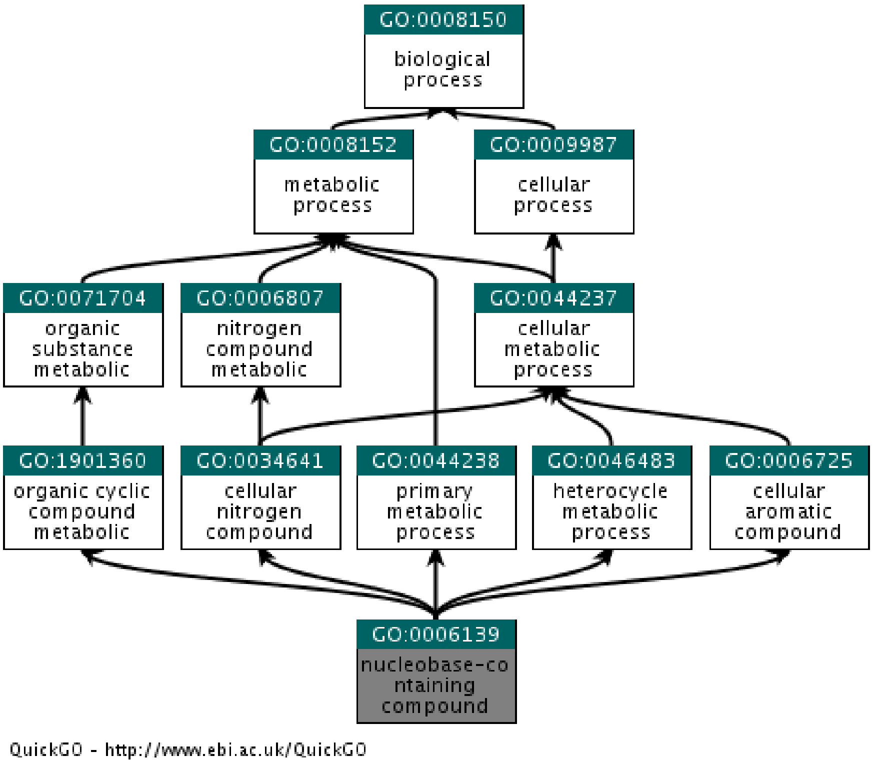

GO:0008150
biological process
GO:0008152
metabolic process
GO:0009987
cellular process
GO:0071704
organic substance metabolic
GO:0006807
nitrogen compound metabolic
GO:0044237
cellular metabolic process
GO:1901360
organic cyclic compound metabolic
GO:0034641
cellular nitrogen compound
GO:0044238
primary metabolic process
GO:0046483
heterocycle metabolic process
GO:0006725
cellular aromatic compound
GO:0006139
nucleobase-containing compound
QuickGO - http://www.ebi.ac.uk/QuickGO

Figure 3.1: Example of a Gene Ontology (GO) sub-hierarchy showing all ancestors of the GO term GO:0006139 (nucleobase-containing compound). This figure was generated using the QuickGO web tool (`http://www.ebi.ac.uk/QuickGO/`).

is 6 levels deep in the FunCat, while in the GO the deepest node is dozens of levels deep.

Figure 3.2 shows a part of the FunCat hierarchy. Note that this hierarchy is structured as a tree, with each node having no or one parent.

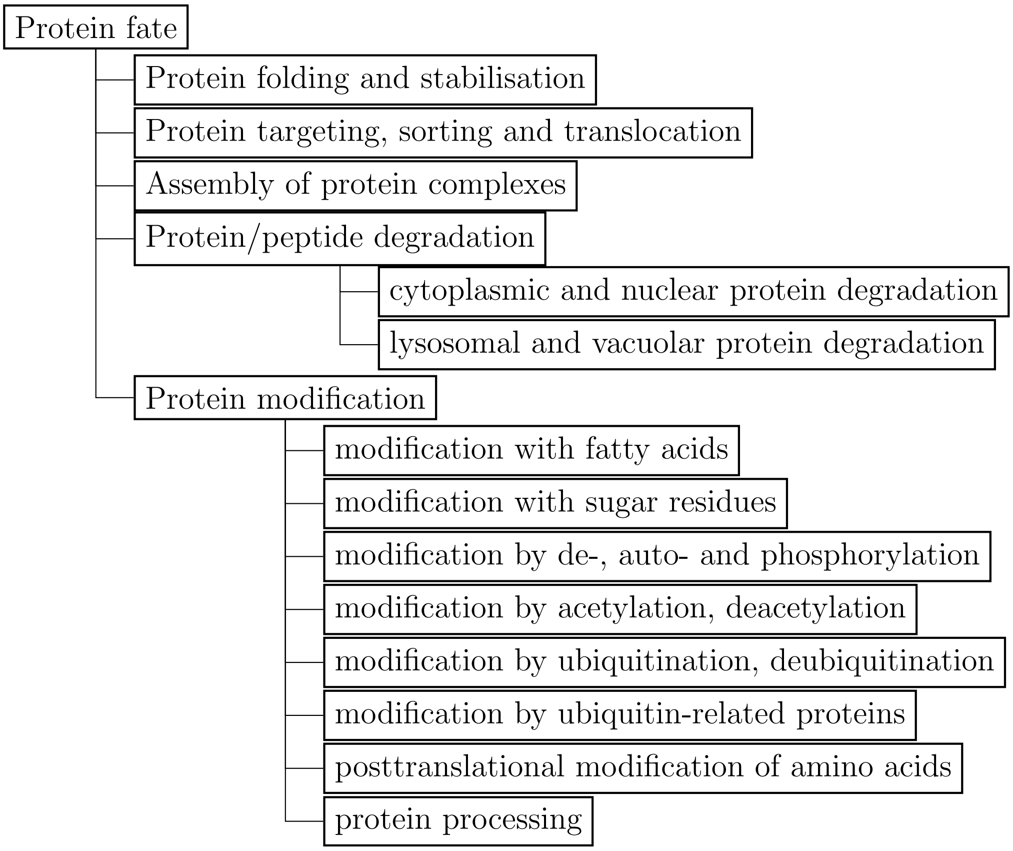

Protein fate
Protein folding and stabilisation
Protein targeting, sorting and translocation
Assembly of protein complexes
Protein/peptide degradation
cytoplasmic and nuclear protein degradation
lysosomal and vacuolar protein degradation
Protein modification
modification with fatty acids
modification with sugar residues
modification by de-, auto- and phosphorylation
modification by acetylation, deacetylation
modification by ubiquitination, deubiquitination
modification by ubiquitin-related proteins
posttranslational modification of amino acids
protein processing

Figure 3.2: Example of a part of the FunCat hierarchy showing all descendants of the term 'protein fate' up to the third level.

3.2 An Overview of Theories of Ageing

Arguably, one of the first steps necessary to investigate the mechanisms of the ageing process is to understand why ageing happens at all (Kirkwood 2002). Among the many existing theories that try to explain the ageing process - Medvedev (1990) catalogued more than 300 - we cite the ones that are the most prominent.

We begin by presenting four popular theories that try to explain the ageing process in the light of evolution.

1. The *disposable soma theory*; first put forward by August Weismann at the

end of the 19th century, and revised by Kirkwood (1977); states that, because of evolutionary pressures, organisms segregate resources between *germ*, reproductive cells, and *soma*, all other cells. As organisms age, they sacrifice the investment in somatic cells to maintain their reproductive capabilities, culminating in the ageing of the organism.

2. The *mutation accumulation theory* (Kirkwood and Austad 2000) puts forth that because the greatest contribution to create the next generation of individuals comes from the young, evolutionary pressure fades as individuals grow older. For this reason, deleterious mutations expressed later in life are not strongly negatively selected.

3. The *antagonistic pleiotropy theory* (Williams 1957) states that there might be advantageous mutations early in life that are harmful later on, and because the selective pressure fades with the ageing of individuals, the negative traits expressed later in life are maintained in the population so that younger individuals can have some evolutionary advantage.

4. The *r- and K-selection* theory is based on the idea that animals in hazardous environments, with high mortality, are selected for rapid development and faster ageing; whilst animals in non-hazardous environments tend to favour delayed development and slower ageing (Austad 1997).

Besides the evolutionary-theories, there are damage-based theories of ageing. These theories state that ageing is the result of accumulative damage in the cellular components of the organism. The *free radical* theory states that cell metabolism generates harmful molecules (reactive oxygen species) that damage cellular components (Harman 1956). However, there is controversy whether this form of ageing is relevant or not. Experiments have been carried out with strains of model organisms with over-expressed genes responsible for combating these molecules and no significant difference in the ageing of individual animals was observed (de Magalhães, Costa and Church 2007). However, oxidation levels in related short-lived and long-lived species were shown to be statistically different (Wieser et al. 2011).

Similarly, the *DNA damage* theory (Szilard 1959) states that the build-up of damage in the DNA could result in ageing, i.e., the damaged DNA may lead to the gradual decline of biological function that we call ageing. This theory has been recently reviewed in (Freitas, Vasieva and de Magalhães 2011).

Although there is controversy whether there are specific genes whose function is to control the ageing process, the idea that single genes influence the ageing process is now broadly accepted by the research community. Several experiments have been carried out to demonstrate the effects of these genes in the ageing process of model organisms. For instance, the knockout of the *age-1* gene in worms results in individuals living twice as long (Friedman and Johnson 1988). In mice, knockout of the *prop-1* gene results in individuals living 50% more (Brown-Borg et al. 1996). One of the most interesting aspects of these experiments is that these genes are involved in the same pathway in the two organisms, which may point to a single genetic mechanism that influences the ageing process in these two completely different species. This motivates the field of bioinformatics and data mining to develop methods for the identification of genes involved in the ageing process.

3.3 A Review of Supervised Machine Learning Applied in Ageing Research

Understanding the ageing process is a very challenging problem in the fields of biology and bioinformatics. Nowadays, with an ever-increasing amount of biological data coming from different high-throughput experiments, it is essential to study this data using machine learning methods that can potentially discover new patterns (or knowledge) in the data, reaching meaningful biological conclusions.

One of the ways machine learning tools can be used to assist biologists understanding the ageing process is through the use of *supervised machine learning algorithms*, which perform classification or regression tasks, as explained in Chapter 2. These algorithms use pre-annotated data, for instance, proteins with known functions, to infer the annotations of new, uncharacterised proteins.

Besides being useful for inference, supervised machined learning algorithms may have the additional purpose of discovering interpretable knowledge. For instance, experts can interpret the results of such algorithms to find patterns to classify a protein as ageing-related, or to investigate the relative importance of features used to predict the chronological age of individuals.

Machine learning experiments are relatively fast; and they can make predictions that help to suggest promising "wet-lab" biological experiments to be done. This approach is cost effective, since wet-lab experiments are in general much slower and expensive than computational experiments.

In this section we review works that use supervised machine learning to study ageing-related proteins and, at the same time, interpret some part of the supervised machine learning results in order to gain biological insights to help understanding the very complex ageing process. This review was done in collaboration with Dr. João Pedro de Magalhães, a biogerontologist at the University of Liverpool, and has been submitted for potential publication (Fabris, Freitas and Magalhães 2017).

3.3.1 A Categorisation of Works on the Biology of Ageing Based on Supervised Learning Tasks

In this section we present a categorisation of papers studying the biology of ageing according to the different types of supervised learning tasks addressed in the papers we reviewed – namely binary classification, hierarchical classification and regression.

The inclusion criteria we adopted were the following: First, the paper must have used a supervised machine learning algorithm during the process of studying the biology of ageing. The work might use the supervised machine learning algorithm as the main source of biological insight (e.g. (Fabris, Freitas and Tullet 2015)) or as an essential part of a larger workflow studying the biology of ageing (e.g. (Huang et al. 2012)). Second, the paper must have discussed at least some part of the predictive model built by the algorithm in the context of the ageing literature. Papers that just report a predictive performance measure for the built model(s), without interpreting it (them), are not the focus of this review.

Ageing is a complex biological phenomenon: it is the result of multiple interacting genetic and environmental factors. Due to this complexity, ageing has been studied at several levels of abstraction using supervised machine learning algorithms, both in the definition of the types of predictor attributes (features) and in the definition of the target variable.

To define predictor attributes, some works use low-level features derived from "raw" amino acid sequences of ageing-related proteins (e.g.: (Fabris and Freitas 2016)). Other works use biomarkers of higher-level biological systems like metabolic and renal systems (e.g.: (Putin et al. 2016)). Some authors even use non-traditional hierarchical features to represent instances, exploring the hierarchical relationships among gene functions available in curated ontologies, such as the Gene Ontology (GO) (e.g.: (Wan and Freitas 2013)).

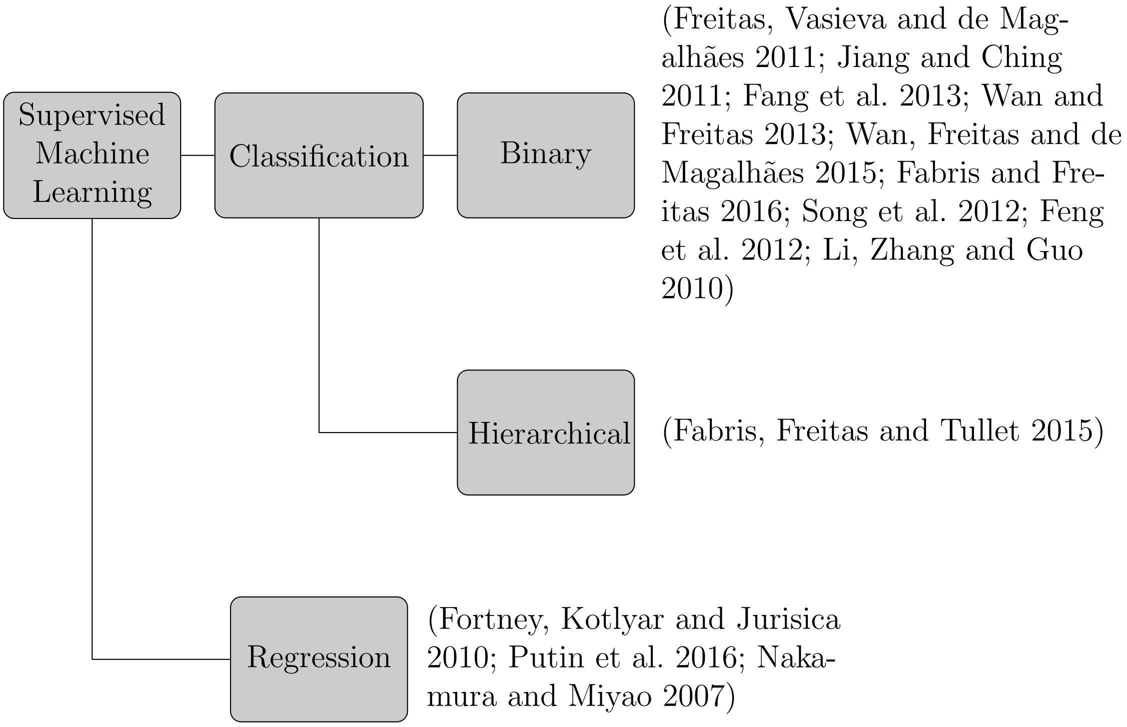

Supervised Machine Learning
Classification
Binary
(Freitas, Vasieva and de Magalhães 2011; Jiang and Ching 2011; Fang et al. 2013; Wan and Freitas 2013; Wan, Freitas and de Magalhães 2015; Fabris and Freitas 2016; Song et al. 2012; Feng et al. 2012; Li, Zhang and Guo 2010)
Hierarchical
(Fabris, Freitas and Tullet 2015)
Regression
(Fortney, Kotlyar and Jurisica 2010; Putin et al. 2016; Nakamura and Miyao 2007)

Figure 3.3: Categorisation of works using supervised machine learning applied to the biology of ageing.

In this work we focus more on the types of target variables used in ageing research, since the type of target variable determines the type of supervised learning task being addressed. Although the use of interpretable predictor attributes is essential for reaching biological conclusions, this topic has been explored in other works about machine learning applied to general biological research (Pandey, Kumar and Steinbach 2006), which could be used as a reference for a biologist using machine learning for studying the ageing problem. On the other hand, a categorisation of the type of target variables to study ageing has never been proposed, as far as we know.

Figure 3.3 shows the full characterisation of the works we considered in the three types of supervised machine learning task (binary classification, hierarchical classification and regression) we are studying. Table 3.1 (in page 67) contains the full list of works that were considered in this thesis with supplementary information about each work. Next, we go into detail on each type of target variable present in the works we reviewed.

Binary Classification in Ageing Research

The majority of works we reviewed uses a binary classification algorithm. Arguably, using binary target variables facilitates interpretation, as the user does not have to deal with the complexities of a larger number of class labels when interpreting the model. It can be argued, however, that some information is lost when not using a larger number of class labels or hierarchical classes (see Section 3.3.1).

When using binary classification, the first task is to define the classes you wish to predict/distinguish between. Next, we list how authors have defined these classes in the works we reviewed.

Ageing-related DNA repair - Some works (Freitas, Vasieva and de Magalhães 2011; Jiang and Ching 2011) have built classification models to allow the discrimination of ageing-related and non-ageing-related DNA repair genes. In these works, the positive class is defined as DNA repair genes that are also related to ageing, while the negative class comprises DNA repair genes that are not related to ageing. This differentiation is important because understanding why some DNA repair genes are ageing-related, while others are not, can help biologists pinpoint the molecular causes or mechanisms of ageing and some progeroid syndromes (accelerated ageing).

In (Fang et al. 2013) authors propose a different but related discrimination, classifying known ageing genes into DNA repair or non-DNA repair related. In other words, the negative class is "ageing-related non-DNA repair", instead of "non-ageing related DNA repair"; while the positive class is the same as in (Freitas, Vasieva and de Magalhães 2011; Jiang and Ching 2011).

Pro-longevity proteins - Other works (Wan and Freitas 2013; Wan, Freitas and de Magalhães 2015) consider pro-longevity vs. anti-longevity class labels when constructing the binary-class datasets. Pro-longevity genes are defined as the genes whose over-expression extends lifespan, or whose decreased activity reduces lifespan. Anti-longevity genes have the opposite effects (Tacutu et al. 2013). This definition of positive and negative instances is interesting to uncover properties that define proteins as pro-longevity or anti-longevity. However, a predictive model built for this binary classification naturally has the weakness that it is not suitable for classifying all proteins of an organism: as the majority of proteins are not pro- nor anti-longevity, models trained without these proteins would likely return incorrect predictions for many proteins with unknown longevity effect.

To address this problem, in (Huang et al. 2012) the authors introduce an additional classifier prior to passing instances to the pro-/anti- longevity classifier. This layer differentiates between lifespan change and no lifespan change. This extra layer complicates model interpretation but enables the use of the classification model in datasets with a larger and more diverse set of proteins.

Another type of target variable definition we have encountered (Li, Dong and Guo 2010) considers as positive "pro-longevity" proteins and as negative proteins that are not "pro-longevity", regardless of whether or not they have an "anti-longevity" effect.

Ageing-related proteins - In (Fabris and Freitas 2016) the authors consider as positive instances proteins that are involved in increased mortality and ageing, and as negative instances proteins that are involved in mortality and *not* involved in ageing. It is interesting to study what differentiates these two classes, since some mutations reduce the lifespan of organisms (e.g., they increase the incidence or lethality of some diseases) but are believed not to be related to ageing.

Hierarchical Classification in Ageing Research

Typical classification problems involve a flat set of class labels, i.e., there are no hierarchical relationships among the class labels to be predicted. By contrast, in hierarchical classification problems, as discussed in Section 2.2, the set of class labels is organised into a hierarchy, usually a tree or a DAG (Directed Acyclic Graph), where each node represents a class label and the edges represent generalisation-specialisation relationships among classes. Dealing with hierarchical classes is common when studying the ageing process, since the main ontology used to annotate proteins related to ageing is the Gene Ontology (GO), which is organised as a DAG where, broadly speaking, nodes represent functions or processes performed by genes or proteins, and edges represent specialisation-generalisation relations between those functions or processes.

Usually authors tend to ignore the hierarchical organisation of the GO and deal only with flat classes, which are easier to interpret and to work with, as traditional classification algorithms can be used. However, hierarchical classification algorithms that exploit hierarchical class relationships can achieve higher predictive performances than "flat" classification algorithms (Silla Jr. and Freitas 2011a).

Recall that hierarchical classification algorithms may be divided into two broad types (Silla Jr. and Freitas 2011a): global or local. Local Hierarchical Classification (LHC) algorithms first build a set of local classification models (base classifiers) by training a traditional (flat) classification algorithm for each (typically small) part of the class hierarchy in the training phase. Then they combine all the local predictions during the testing phase, when predicting the class of a new instance. By contrast, global hierarchical classification algorithms build a single global classification model predicting classes in the whole class hierarchy.

Global hierarchical classification algorithms have the advantage of producing a single coherent global classification model, which tends to be more easily interpreted than a large number of different local classification models. For more details about the differences between local and global approaches for hierarchical classification, please refer to Section 2.2.

The work of Fabris, Freitas and Tullet (2015), the only one to deal with hierarchical classification of ageing-related genes/proteins and interpreting the corresponding classification model, uses a global hierarchical decision tree model to classify ageing-related proteins in hierarchical classes. The classes are ageing-related because they are the over-expressed hierarchical classes present in the ageing-related proteins from the GenAge database (de Magalhães et al. 2009).

Regression in Ageing Research

Some works use regression techniques to study the ageing problem. Recall that in regression the target variable is continuous (real-valued) whilst in classification problems the target variable is categorical (nominal or discrete).

In ageing research, regression techniques have been used to predict chronological age given a set of biomarkers (Fortney, Kotlyar and Jurisica 2010; Putin et al. 2016) and to build an index of the rate of ageing given a set of biomarkers (Nakamura and Miyao 2007).

It is important to identify which biomarkers are most related to ageing phenotypes, thus enabling, for instance, the use of biomarkers to measure the results of interventions in ageing-research.

3.3.2 Biological Insights Derived from the Supervised Machine Learning Algorithms

Supervised Machine Learning Findings Support the Link Between Ageing/Longevity and Specific Types of DNA Repair

The link between DNA repair and ageing/longevity is well established in the biological literature: it has been shown that some ageing-related diseases in humans are directly linked to malfunctioning pathways related to DNA maintenance – e.g. some progeroid (accelerated ageing) syndromes are caused by mutations in DNA repair (Lombard et al. 2005; Freitas and de Magalhães 2011). Moreover, it has been shown that over-expression of DNA-repair-related genes increase lifespan in some animal species and that DNA-repair efficiency is positively correlated with increased longevity in several species (Shaposhnikov et al. 2015).

In (Freitas, Vasieva and de Magalhães 2011), the authors noted, after analysing the model generated by the decision tree algorithm J48 (induced to differentiate between ageing-related DNA repair genes and non-ageing-related DNA repair genes in humans), that if a DNA repair gene's protein product interacts with XRCC5 (Ku80), that gene is likely ageing-related.

Interestingly, links between Ku proteins and longevity have been found by other supervised machine learning works studying connections between DNA repair genes and ageing in humans. In (Jiang and Ching 2011), the authors use an SVM algorithm to distinguish between human ageing-related DNA repair genes and human non-ageing-related DNA repair genes. By analysing the instances (proteins) furthest from the SVM's hyperplane, the authors identified that XRCC6 (Ku70) and MLH1 are strongly predicted as ageing-related. Ku70, Ku80 (from (Freitas, Vasieva and de Magalhães 2011)) and MLH1 are involved in non-homologous end joining (Bannister, Waldman and Waldman 2004; Fattah et al. 2010). Interestingly, the Ku protein family is highly conserved among eukaryote species and is a well-conserved longevity regulator across species, being a key target of ageing research (Dynan and Yoo 1998). The authors also point out that PARP1, PCNA and APEX1 are essential to base excision repair, a pathway that is known to be affected by deficient WRN proteins, which are directly linked to Werner syndrome, a disease characterised in humans by accelerated ageing.

Not surprisingly, a homolog of WRN (WRN1) has been identified as longevity-related in the worm while classifying worm genes into longevity-related and non-longevity-related in (Li, Dong and Guo 2010). Interestingly, however, defects in the WRN protein in mice do not cause, by themselves, Werner-like phenotypes. However, in conjunction with defects in P53 they do cause the typical Werner syndrome phenotypes (Lombard et al. 2000). This stresses the already known fact that even in relatively closely related species (like mouse and human) ageing-related genes in one species may not directly lead to the same ageing phenotype in another (de Magalhães 2014).

Still regarding WRN and P53, in (Fabris, Freitas and Tullet 2015) the authors noted (by interpreting a classification model) that interaction with P53 is a good predictor of ageing-related GO terms in humans and mouse. In both human and mouse, P53 is closely related to WRN, and both proteins participate in DNA repair, reinforcing the importance of WRN to predict ageing-related GO terms in humans.

In (Wan, Freitas and de Magalhães 2015) one of the features selected to classify proteins as ageing-related in yeast was the GO term "double-strand break repair", directly related to DNA-repair.

In (Huang et al. 2012), "mitochondria genome maintenance", also related to DNA maintenance, was one of the features selected to predict the effect of a gene deletion as a "change" or "no change" of lifespan in yeast.

In (Fang et al. 2013), the authors used the Gini index calculated by the Random Forest algorithm to select the 18 most relevant Protein-Protein-Interaction (PPI) features. Out of these 18 features, the authors highlighted interactions with proteins BLM, ERCC2, FANCG, MSH2, ATM, MRE11A and ATR, which play a role in check-point control and DNA damage check; and interactions with proteins BLM, WRN, MRE11A and Mre11, which are associated with the maintenance of telomeres (Fang et al. 2013).

In summary, it appears that the results of supervised machine learning algorithms have corroborated the fact that DNA repair is strongly linked to ageing/-longevity. DNA repair-related features are commonly chosen as good predictors of ageing/longevity by classification algorithms. Furthermore, proteins related to DNA-repair and maintenance are commonly predicted as ageing-related. To some extent, the machine learning algorithms are reflecting a bias stemming from the biological knowledge already encoded in the data. However, the algorithms can also find proteins highly related with DNA repair that are not annotated as ageing-related (like the ones studied in (Li, Dong and Guo 2010)). In fact, in (Li, Dong and Guo 2010), the authors proceeded to carry out wet-lab experimentation of proteins predicted to be longevity-related in worms, identifying two proteins that increase lifespan in the animal, namely: VPS-34 and PHI-38. In addition, the analysis of the classification models built by the algorithms suggests that, out of the different types of DNA repairs, non-homologous end joining seems to be the one most relevant for the ageing process.

Ageing-Related Proteins Tend to be Highly Connected and Are Enriched for Certain Functions

Other important type of biological conclusion derived by supervised machine learning algorithms is how ageing-related proteins are connected both between themselves and with non-ageing related proteins.

In (Li, Dong and Guo 2010), the authors conclude (with statistical support) that several proteins' properties derived from interaction graphs (topological features) and sequence analysis have different distributions, depending on whether or not they are longevity-related. Next, we summarise the main biological conclusions the authors have drawn for longevity-related proteins or genes in worms:

1) Longevity-related proteins (LRPs) have, on average, more interaction partners than non-LRPs. 2) LRPs have more LRPs in their neighborhood than non-LRPs. 3) LRPs tend to be closer between themselves than to other proteins. 4) LRPs tend to occupy a more central location in the PPI graph. 5) LRPs tend to be expressed together. 6) LRPs tend to have longer amino acid sequences. 7) Longevity-related genes tend to be more conserved. 8) Longevity-related genes are enriched for certain functions (e.g. Negative regulation of cell proliferation, Positive regulation of non-apoptotic programmed cell death, Cilium biogenesis and regulation). 9) Longevity-related genes tend to show certain RNAi phenotypes (e.g. abnormal DAF dauer formation, abnormal transgene expression, and variable embryonic terminal arrest).

Similarly to (Li, Dong and Guo 2010), in (Song et al. 2012; Feng et al. 2012; Li, Zhang and Guo 2010) the authors use topological network features to study the ageing process. The latter works, however, predict ageing-related genes vs. non-ageing-related genes, not longevity vs. non-longevity genes. The difference between non-ageing-related genes and longevity-related genes is subtle but important: an increase in longevity may not be a result of changes in the ageing process. For instance, longevity may be increased by mutations that improve resistance against some disease, not altering the overall ageing process (de Magalhães et al. 2009). Also, the works in (Song et al. 2012; Feng et al. 2012; Li, Zhang and Guo

2010), study fruit flies, mice and humans, while (Li, Dong and Guo 2010) studies worms. Next, we summarise the biological conclusions drawn in (Song et al. 2012; Feng et al. 2012; Li, Zhang and Guo 2010); all with statistical support.

1) For fly, mouse, and human; ageing-related proteins (ARPs) have, on average, more interaction partners than non-ageing-related proteins. 2) For fly and mouse; ARPs tend to be in more tightly connected protein clusters than non-ARPs. 3) For fly, mouse, and humans; ARPs have more ARPs in their neighborhood than non-ARPs. 4) For fly; ARPs tend to be closer between themselves than to other proteins. 5) For fly and human; ARPs tend to occupy a more central location in the PPI graph. 6) For fly and mouse; ARPs behave more like hubs than non-ageing related proteins. 7) For human; ARPs tend to be expressed together. Hence, despite the differences in species and data, the conclusions in (Song et al. 2012; Feng et al. 2012; Li, Zhang and Guo 2010) are broadly similar to the conclusions in (Li, Dong and Guo 2010).

In (Fortney, Kotlyar and Jurisica 2010), the authors show that their technique of creating modular subnetworks of longevity genes creates modules with statistically significantly more longevity genes. Also, they claim, with statistical support, that modular subnetworks participate in many different age-related biological processes.

In (Huang et al. 2012), the authors indicate that the edge density and edge weight density of the deletion network and the local centrality of a deletion gene can be used to predict ageing-related effects of gene deletion.

In summary, all these works point to the fact that ageing-related proteins tend to be more connected than other proteins and also appear to be closer to each other in protein-protein interaction networks than other proteins.

Autophagy and Apoptosis Mechanisms Are Associated with Ageing and Longevity

The mechanisms of autophagy and apoptosis control the turnover of organelles and programmed cell death, respectively. Several works have identified that autophagy/apoptosis-related proteins are related to ageing (Kurz, Terman and Brunk 2007). Next, we review the main biological conclusions regarding these processes drawn from supervised machine learning works.

In (Li, Zhang and Guo 2010), authors classify human proteins as ageing-related or non-ageing-related using SVMs. They highlight that protein “VPS-34”, involved in autophagy (Jaber and Zong 2013), was predicted as ageing-related with a probability of 0.93.

In (Feng et al. 2012), where authors classify mouse proteins as ageing-related or non-ageing-related using SVMs, they mention that the gene Akt1 was predicted as ageing-related with high probability, and this gene is strongly involved in apoptosis (Xu, Liu and Songyang 2002).

In (Wan and Freitas 2013), among the top ranking terms selected to predict the pro-longevity effect of a gene in worms were “autophagy”, and “apoptotic process”, once again reinforcing the link between ageing and autophagy/ apoptosis. In (Wan, Freitas and de Magalhães 2015), the authors identified “apoptotic signaling pathway” as a good anti-longevity predictor for worms.

In (Fang et al. 2013), interactions with a number of proteins, including BLM, ATM, RPA1, PCNA and HSPA4, which are linked to the apoptosis of tumor cell lines, were found to be good predictors of DNA-repair ageing-related proteins.

In (Fabris and Freitas 2016), for mouse, it was found that if a protein has any influence on CDK1 (associated with apoptosis), then that protein is more likely to be ageing-related. CDK1 is putatively associated to ageing in mice (Xiao et al. 1999).

In summary, several works have identified that proteins related to apoptosis and autophagy are good predictors of ageing-relatedness. This is in agreement with the literature, as these mechanisms are essential to tumor suppression and regulation of oxidative stress (Filomeni, De Zio and Cecconi 2015), which in turn, are linked to the ageing process.

Interactions with Nutrient Receptor Genes are Good Predictors of Ageing-Relatedness

The link between nutrient sensing and ageing is well established in several organisms: mutations in genes responsible for nutrient signalling and food recognition promote longevity across species (López-Otín et al. 2016). The following works identified protein features (functional annotations) related to nutrient signalling that are good predictors of a role in the ageing process.

In (Fabris, Freitas and Tullet 2015), the authors identify that in fruit flies some over-represented GO terms are involved in food recognition. The work in (Wan, Freitas and de Magalhães 2015) shows that in fruit flies, “sensory perception”, which is essential to food recognition, was among the top selected features using their feature selection algorithm.

The work in (Wan and Freitas 2013), studying worms, shows that one of the top features selected by their feature selection algorithm is "response to nutrient level".

In summary, not surprisingly, proteins involved in nutrient sensing have been found to be good predictors of ageing-relatedness, since the connection between nutrient sensing and ageing is well-known in fruit flies and worms (Kapahi et al. 2010).

3.3.3 Other Conclusions Reported in the Literature

Copper and Iron Ion Transport

It has been shown that deficiencies in iron and copper levels are linked with ageing-related diseases like Atherosclerosis and Alzheimer's disease (Brewer 2007). It is interesting that two papers using supervised machine learning algorithms have found that interactions with proteins involved in copper and iron ion transport are good predictors of ageing/longevity, as follows.

In (Song et al. 2012), atp7 was predicted as ageing-related in fruit flies. This gene is involved in copper ion transport in fruit flies (Norgate et al. 2006).

The gene ftn-1 (related to iron ion transport) was predicted (with a probability of 0.95) as a longevity gene in (Li, Dong and Guo 2010). It has been shown that the lack of ftn-1 caused a reduced lifespan when worms are under iron stress (Kim et al. 2004).

Comparing Findings from Machine Learning and Functional Enrichment Analysis

One recent study investigated pathways overrepresented in pro- and anti-longevity genes using both traditional functional enrichment analysis and a machine-learning-based feature selection method (Fernandes et al. 2017). Although the two methods work in different ways, some overlap between their results was observed. For example, both methods found terms related to insulin signaling or growth to be significantly associated with anti-longevity genes in mice; and terms related to autophagy were found to be significantly associated with pro-longevity genes in *C. elegans* by both methods. Therefore, it seems that both functional enrichment and machine learning algorithms identified the major, most significant pathways associated with longevity genes. However, there were also many pathways that were only identified by each of the methods (Fernandes et al. 2017). Hence, both methods provide partially complementary information.

Biomarkers

Besides the ageing/longevity-related studies at the genomic/proteomic level, we have identified works that use physiological markers to study the ageing process. Next, we discuss two works following such approach.

In (Putin et al. 2016), the authors claim that the analysis of relative feature importance within deep neural networks trained to predict chronological age on humans helped deduce the most important features that may shed light on the contribution of individual biological systems to the ageing process. The systems were ranked in the following order of decreasing importance, according to the selected features: metabolic, liver, renal system and respiratory function. The ranking was created by counting the number of biomarkers coming from the different systems.

In (Nakamura and Miyao 2007), the authors investigate good predictors of chronological age of humans to develop an expression to calculate the biological age score of individuals. The authors used a population of 86 men, which were evaluated annually during six years. Their chronological age (the target variable) and the result of physiological exams (the predictive variables) were recorded and used in this study. Using Principal Components Analysis (PCA) and logistic regression, the authors identified the following five candidate biomarkers of ageing for constructing the score of biological age: systolic blood pressure, forced expiratory volume in 1 second divided by height squared, hematocrit blood level, albumin blood level, and blood urea nitrogen level.

3.3.4 Summary of Data Mining Findings About Ageing Biology

We can conclude, based on our analysis of the literature using supervised machine learning applied to ageing research, that several already known biological facts were corroborated by supervised machine learning algorithms. Namely, it was found that interactions with DNA repair genes and proteins are good predictors of ageing-relatedness, in special DNA repair proteins closely related to the Ku protein family (Freitas, Vasieva and de Magalhães 2011; Jiang and Ching 2011), the WRN protein (Jiang and Ching 2011; Li, Dong and Guo 2010; Fang et al. 2013) and the P53 protein (Fabris, Freitas and Tullet 2015).

We have also observed that several works (Li, Dong and Guo 2010; Song et al. 2012; Feng et al. 2012; Li, Zhang and Guo 2010) concluded that ageing-related proteins tend to be highly connected and seem to play a central role in molecular pathways. Additionally, works link ageing/longevity with autophagy and apoptosis (Li, Zhang and Guo 2010; Feng et al. 2012; Wan and Freitas 2013; Wan, Freitas and de Magalhães 2015; Fang et al. 2013; Fabris and Freitas 2016); nutrient receptor genes (Fabris, Freitas and Tullet 2015; Wan, Freitas and de Magalhães 2015; Wan and Freitas 2013); and copper and iron ion transport (Song et al. 2012; Li, Dong and Guo 2010).

From a higher-level perspective, several biomarkers were found to be ageing related. In (Putin et al. 2016), biomarkers in the following systems were identified: metabolic, liver, renal system and respiratory function. In (Nakamura and Miyao 2007), the authors highlighted systolic blood pressure, forced expiratory volume in 1 second divided by height squared, hematocrit blood level, albumin blood level, and blood urea nitrogen level.

Unfortunately, predictions of classification algorithms were only experimentally confirmed in one paper (Li, Dong and Guo 2010). We believe that a stronger integration between machine learning and wet-lab experimentation would improve the application of interpretable machine learning algorithm in ageing research. Experimental confirmation of *in silico* predictions is important to validate the conclusions of machine learning algorithms, from a biological perspective.

Table 3.1: Main data analysis characteristics of papers that focus on applying some supervised machine learning algorithm to tackle a biological ageing problem and then interpret the results to get some type of biological insight about the ageing process. Columns 1 and 4 to 6 inform us, respectively, the type of classification problem that was considered in the paper, the supervised learning algorithm whose results were interpreted, the feature type used by the algorithm and the species that were considered in the interpretation. Note that a paper may contain other machine learning algorithms, feature types and species whose results were not interpreted and therefore are not listed in the table.

Type of Supervised Machine Learning Problem	Ref.	Paper's Title	Supervised Learning Algorithm	Feature Type	Species
Binary classification problem (involving DNA repair and ageing-related proteins)	Freitas, Vasieva and de Magalhães (2011)	A data mining approach for classifying DNA repair genes into ageing-related or non-ageing-related	Decision Tree (J48)	Protein-protein interactions (PPI), Gene Expression, Gene Ontology terms, type of DNA Repair, Dn/Ds ratio	Human
	Jiang and Ching (2011)	Classifying DNA repair genes by kernel-based support vector machines	SVM (Support Vector Machine)	Gene expression levels	Human
	Fang et al. (2013)	Classifying Aging Genes into DNA Repair or Non-DNA Repair-Related Categories	Feature Selection Based on Random Forests	Protein-protein interactions	Human

Binary classification using hierarchical features (pro-longevity vs. anti-longevity proteins)	Wan and Freitas (2013)	Prediction of the pro-longevity or anti-longevity effect of *Caenorhabditis Elegans* genes based on Bayesian classification methods	Hierarchical Feature Selection used in the first phase of a naive Bayes algorithm	Gene Ontology terms	Worm
	Wan, Freitas and de Magalhães (2015)	Predicting the pro-longevity or anti-longevity effect of model organism genes with new hierarchical feature selection methods	Hierarchical Feature Selection used in the first phase of a naive Bayes algorithm	Gene Ontology terms	Worm, fly, mouse, yeast
Hierarchical classification (using proteins as instances and ageing-related GO terms as classes)	Fabris, Freitas and Tullet (2015)	An Extensive Empirical Comparison of Probabilistic Hierarchical Classifiers in Datasets of Ageing-Related Genes	Decision Tree for hierarchical classification	Protein-protein interactions	Worm, fly, mouse, human, yeast
Binary classification (ageing-related vs. non-ageing-related mortality-related proteins)	Fabris and Freitas (2016)	New KEGG pathway-based interpretable features for classifying ageing-related mouse proteins	Decision Table	KEGG pathway features	Mouse

Binary classification (ageing-related vs. non-ageing-related genes)[2]	Song et al. (2012)	Discovering aging-genes by topological features in *Drosophila melanogaster* protein-protein interaction network	SVM	PPI Network features	Fruit fly
	Feng et al. (2012)	Topological analysis and prediction of aging genes in *Mus musculus*	SVM	PPI Network features	Mouse
	Li, Zhang and Guo (2010)	Computational prediction of aging genes in human	SVM, k-NN, Decision Tree	PPI Network features	Human
Binary classification (Longevity vs. non-longevity genes)	Li, Dong and Guo (2010)	Systematic analysis and prediction of longevity genes in *Caenorhabditis elegans*	SVM, k-NN, Decision Tree	Functional interaction network features, conservation score	Worm
Two-layer binary classification (life span change and then increase or decrease the life span of genes)	Huang et al. (2012)	Deciphering the effects of gene deletion on yeast longevity using network and machine learning approaches	Selected features using k-NN with Incremental feature selection	PPI Network, biochemical, physicochemical, functional, and deletion features	Yeast
Regression (prediction of chronological age)	Fortney, Kotlyar and Jurisica (2010)	Inferring the functions of longevity genes with modular subnetwork biomarkers of *Caenorhabditis elegans* aging	SVR (support vector regression)	Modular features from gene interaction networks	Worm
Regression (prediction of rate of ageing)	Nakamura and Miyao (2007)	A method for identifying biomarkers of aging and constructing an index of biological age in humans	Logistic Regression	Various biomarkers	Human

[2] These works also include analysis of individual features

Chapter 4

A More Efficient Local Hierarchical Classification Algorithm

This chapter presents a modified version of the Extended Local Hierarchical Naive Bayes algorithm, which exploits the requirements of the original algorithm (single-path, mandatory-leaf-prediction hierarchical classification problems in tree-structured class hierarchies) to greatly improve classification run-time. We show that, considering 18 hierarchical classification datasets, the modified algorithm yields equivalent predictive performance and significantly improves run-time in the training and prediction phases, by comparison with the original algorithm.

The modified algorithm proposed in this chapter was first introduced in (Fabris and Freitas 2014a).

4.1 Introduction

This chapter focuses on the Extended Local Hierarchical Naive Bayes (ELHNB) classification algorithm, recently proposed in (Merschmann and Freitas 2013). In that work, a Naive Bayes (NB) local classifier was trained for each class label considering the class labels of the neighbouring nodes (parent and children) in the class hierarchy as extended features. In the prediction phase, the algorithm marginalises out the extended features by summing up the probabilities of all possible combinations of neighbouring classes. This algorithm is classified as "extended local" because although they use local NB classifiers, the class hierarchy is taken into consideration by using the extended features.

The objective of using classification algorithms in this chapter is to infer protein functions using only attributes (or features) describing the protein and gene sequence information. That is, to build a function $\mathcal{F}$ that maps the gene and protein sequences to class labels, namely, their functional classes, their cellular location, molecular function, and the biological processes that they are involved in. The inferred class labels may be used as a starting point to select the most promising wet-lab experiments to be performed, which are much more time-consuming and expensive than their computational counterparts.

However, unlike the "flat" class labels of standard classification problems, ontologies that define the classification of genes and proteins are normally organised as trees or Directed Acyclic Graphs (DAGs). This complicates the use of out-of-shelf classification algorithm and justifies the development of specialised algorithms for hierarchical classification (Silla Jr. and Freitas 2011a).

Additionally, due to the exponentially like ever-increasing size of biological datasets (Stein 2003; The UniProt Consortium 2014), it is important to develop efficient algorithms that scale up well both in the training and prediction phases. This factor has been neglected in the literature, where works containing complexity and/or run-time analysis of hierarchical classification algorithm are uncommon. To help fill this niche, this work proposes a modified version of the Extended Local Hierarchical Naive Bayes (ELHNB) originally proposed by Merschmann and Freitas (2013). The modification (M-ELHNB) speeds up the training and prediction phases of the algorithm, maintaining a statistically equivalent predictive performance in general.

4.2 Extended Local Hierarchical Naive Bayes

Applying traditional ('flat') classifiers in hierarchical classification problems is not straightforward. Although the literature has multiple works on the subject of applying traditional algorithms in the hierarchical classification setting (Silla Jr. and Freitas 2011a), algorithms that explicitly take the class hierarchy into account have the potential of improving the classification performance, reducing model size, and improving the interpretability of the classifier (Blockeel et al. 2006).

For these reasons, one of the successful efforts towards creating specific classification algorithms for hierarchical classification is the work of Merschmann and

Freitas (2013), which is an algorithm specially tailored for hierarchical classification that takes into account the class hierarchy explicitly. This algorithm, however, has a specially lengthy prediction phase due to the necessity of various probability calculations. We propose a modification of the algorithm that significantly improves its run time, specially in the prediction phase.

4.2.1 The Original Algorithm

The ELHNB algorithm was designed specifically for $\langle T, SPL, FD\rangle$ hierarchical classification problems, i.e., classification problems with Tree-organised (T) class label taxonomies, with every instance having a Single Path of Labels (SPL), with Full Depth (FD) class labels.

This algorithm estimates the probability of an instance belonging to each class label C_i given its features $\mathbf{x}$ using eq. (4.1) (readers interested in the derivation may refer to the original paper.) Eq. (4.1) is used for each class label $C_i, 1 \leq i \leq N$, where N is the number of class labels in the hierarchy.

$$\begin{aligned} P(C_i = c_i|\mathbf{x}) = \frac{1}{P(\mathbf{x})}\times \\ \sum_{\mathbf{y_i}\in\{0,1\}^{k(i)}} \left(\frac{P(\mathbf{x}|C_i = c_i, \mathbf{y_i})P(C_i = c_i|\mathbf{y_i})P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i})}{\sum_{c'\in\{c_i,\tilde{c_i}\}} P(\mathbf{x}|C_i = c', \mathbf{y_i})P(C_i = c'|\mathbf{y_i})}\right) \end{aligned} \tag{4.1}$$

In Eq. (4.1) $k(i)$ represents the number of neighbours of the i-th class label (the set containing the children and parent of the i-th class node, given by the hierarchy), $\mathbf{y_i}$ is a binary vector that iterates over all possible classifications of the neighbourhood of the i-th class label, C_i is a random variable that may take the values c_i and $\tilde{c_i}$, $C_i = c_i$ is the event of the current instance being classified as belonging to the i-th class label and $C_i = \tilde{c_i}$ the event of the current instance being classified as not belonging to the i-th class.

To estimate $P(\mathbf{x}|C_i, \mathbf{y_i})$ and $P(\mathbf{x}|\mathbf{y_i})$, the authors use the Naive Bayes assumption that the predictive attributes are independent given the class, reducing the estimation expressions to $P(\mathbf{x}|C_i, \mathbf{y_i}) = \prod_{k=1}^{n} P(x_k|C_i, \mathbf{y_i})$ and $P(\mathbf{x}|\mathbf{y_i}) = \prod_{k=1}^{n} P(x_k|\mathbf{y_i})$, where n is the number of predictive features. The probabilities $P(C_i|\mathbf{y_i})$ and $P(\mathbf{y_i})$ may be estimated by simple counting and $P(\mathbf{x})$ is computed by using the fact that $P(C_i = c_i|\mathbf{x}) + P(C_i = \tilde{c_i}|\mathbf{x}) = 1$.

The probabilities calculated by the algorithm are not guaranteed to be consistent, that is, the probability of an instance $\mathbf{x}$ belonging to class label i, $P(C_i = c_i|\mathbf{x})$, may be higher than the probability of the instance belonging to the parent of i, which is inconsistent to the fact that if an instance belongs to class label i it implicitly belongs to the parent of i. Therefore, to tackle this problem, the authors calculate the geometric average of the probabilities of all class labels in each of the possible paths from the root to the leaves and choose the path with greatest average class label probability as the final classification, bypassing the inconsistency problem.

The overall time complexity of the algorithm's training phase considering the number of probability values that need to be estimated is $O(S_{mean} \times N)$, where S_{mean} is the mean size of the neighborhood of all class nodes and N is the number of class nodes. Naturally, the complexity of the prediction phase considering the number of evaluations of probability values is also $O(S_{mean} \times N)$.

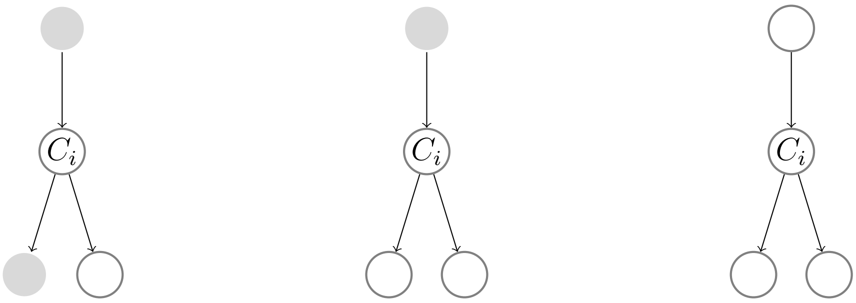
4.2.2 The Modified Algorithm

We shall demonstrate that for problems classified as $\langle T, SPL, FD\rangle$, the training phase of the Modified Extended Local Hierarchical Naive Bayes (M-ELHNB) algorithm may be executed faster by reducing the number of probability values that must be estimated. Likewise, we shall demonstrate that the prediction phase (the estimation of $P(C_i = c_i|\mathbf{x})$) for a given class label may be executed in constant time complexity regarding the number of its child nodes in the class hierarchy. Since C_i is a binary variable, it is enough to compute either $P(C_i = c_i|\mathbf{x})$ or $P(C_i = \tilde{c}_i|\mathbf{x})$. Next we show how to compute $P(C_i = \tilde{c}_i|\mathbf{x})$ since it allows for more simplifications in Equation (4.1).

Theorem 4.1. *Estimating $P(\mathbf{x})P(C_i = \tilde{c}_i|\mathbf{x})$ requires constant time complexity (that is, the estimation of time complexity is independent of the number of neighbours of class label C_i in the class hierarchy).*

First, let us consider non-leaf nodes, we may decompose Eq. 4.1 as follows:

$$P(C_i = \tilde{c}_i|\mathbf{x}) = \frac{1}{P(\mathbf{x})}\times$$
$$\left[\sum_{\mathbf{y_i}\in S_1}\left(\frac{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i})P(C_i = \tilde{c}_i|\mathbf{y_i})P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i})}{\sum_{c'\in\{c_i,\tilde{c}_i\}} P(\mathbf{x}|C_i = c', \mathbf{y_i})P(C_i = c'|\mathbf{y_i})}\right)+ \right. \tag{4.2}$$



(a) Case 1 - C_i's parent is active and exactly one child node of C_i is active. In this case, the number of possible configurations of labels for the set y_i of neighbours of C_i is equal to the number of children of C_i.

(b) Case 2 - C_i's parent is active, C_i's children are not active. This case contains only one possible configuration of labels for the set y_i of neighbours of C_i.

(c) Case 3 - None of the neighbours of C_i is active. This case contains only one possible configuration of labels for the set y_i of neighbours of C_i.

Figure 4.1: Simplified examples of the three possible types of label configurations for the neighbourhood of a class node. Shaded nodes represent the active neighbourhood of the node C_i, i.e., neighbours whose class label is present in an instance.

$$\sum_{\mathbf{y_i} \in S_2} \left(\frac{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i})P(C_i = \tilde{c}_i|\mathbf{y_i})P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i})}{\sum_{C_i \in \{c_i, \tilde{c}_i\}} P(\mathbf{x}|C_i = c', \mathbf{y_i})P(C_i = c'|\mathbf{y_i})} \right) + \tag{4.3}$$

$$\sum_{\mathbf{y_i} \in S_3} \left(\frac{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i})P(C_i = \tilde{c}_i|\mathbf{y_i})P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i})}{\sum_{C_i \in \{c_i, \tilde{c}_i\}} P(\mathbf{x}|C_i = c', \mathbf{y_i})P(C_i = c'|\mathbf{y_i})} \right) \Bigg]. \tag{4.4}$$

Class label sets S_1, S_2 and S_3 – in expressions (4.2), (4.3) and (4.4), respectively – are the only three possible types of label configuration for the neighbourhood of the i-th class node: 1) one parent and one child node, 2) one parent and no children and, 3) no parents and no children. Notice that these are the only possibilities because we are dealing with single-path predictions in a class-tree setting. Figure 4.1 displays a graphical representation of the three possibilities.

We may eliminate expression (4.2) from the summation since, when $\mathbf{y_i} \in S_1$, $P(C_i = \tilde{c}_i|\mathbf{y_i})$ equals to 0 because there is no case where $\tilde{c}_i$ happens and both its parent and some child node are active (have a positive class label). This is because, due to the "Is-a" hierarchy, if a child of C_i is active, C_i must be active too. Similarly we may simplify expressions (4.3) and (4.4) considering that $P(C_i = c_i|\mathbf{y_i})$

is equal to 0 when $\mathbf{y_i} \in S_2$ *or* $\mathbf{y_i} \in S_3$*. Equation* (4.7) *presents the simplifications.*

$$P(C_i = \tilde{c}_i|\mathbf{x}) = \frac{1}{P(\mathbf{x})}\times \tag{4.5}$$

$$\left[\sum_{\mathbf{y_i}\in S_2}\left(\frac{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i})P(C_i = \tilde{c}_i|\mathbf{y_i})}P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i})}{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i})P(C_i = \tilde{c}_i|\mathbf{y_i})}}\right)+\right.$$

$$\left.\sum_{\mathbf{y_i}\in S_3}\left(\frac{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i})P(C_i = \tilde{c}_i|\mathbf{y_i})}P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i})}{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i})P(C_i = \tilde{c}_i|\mathbf{y_i})}}\right)\right] \tag{4.6}$$

$$P(\mathbf{x})P(C_i = \tilde{c}_i|\mathbf{x}) =$$

$$\sum_{\mathbf{y_i}\in S_2} P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i}) + \sum_{\mathbf{y_i}\in S_3} P(\mathbf{x}|\mathbf{y_i})P(\mathbf{y_i}). \tag{4.7}$$

Therefore, because sets S_2 *and* S_3 *are of unitary cardinality, containing only one possible configuration of labels for the neighbour (parent or child) classes of* C_i*, calculating* $P(\boldsymbol{x})P(C_i = \tilde{c}_i|\boldsymbol{x})$ *requires constant time complexity with respect to the number of neighbours of* C_i *in the class hierarchy.*

To calculate the probability $P(\boldsymbol{x})P(C_i = \tilde{c}_i|\boldsymbol{x})$ *for leaf nodes we may follow the same steps considering that the set* S_1 *is empty (the leaf nodes have no children), the set* S_2 *contains only one element (the parent node of the leaf) and the set* S_3 *is empty, meaning that the time complexity for the probability estimation for leaf nodes is also independent of the number of neighbours.* ■

To estimate the normalising constant $P(\mathbf{x})$ we would need the value of $P(C_i = c_i|\mathbf{x})$, which is the original probability estimated by Merschmann and Freitas (2013), the very value that we would like to avoid calculating because of the time complexity dependence on the number of neighbours of the class node.

Thus, we use a heuristic to get rid of the normalising constant $P(\mathbf{x})$: we calculate all non-normalised probabilities and assume that the largest one, $Z_{pseudo} = \max_{C_i}(P(\mathbf{x})P(C_i = \tilde{c}_i|\mathbf{x}))$, is the pseudo-normalisation constant. Additionally, we do not use the geometric average approach proposed by Merschmann and Freitas (2013) to find the most probable path in the class hierarchy; instead, we use a top-down strategy: first find the most probable class label that is a child of the root node, then, recursively find the most probable child of this node, and so on, until a leaf node is reached. This approach produced better results than the original strategy of using the geometric average in our initial experiments.

The overall time complexity of the training phase of the modified algorithm, considering the number of probability values that need to be estimated, is $O(N)$, where N is the number of class nodes in the hierarchy. The time complexity of the prediction phase, considering the number of evaluations of probability values is also $O(N)$. In other words, the runtime complexity of our modified algorithm does not depend on the topology of the class hierarchy (how the classes are connected), just on the number of classes.

Both time complexities are independent of the number of neighbours of the class nodes and are strictly smaller than $O(S_{mean} \times N)$, where S_{mean} is the mean number of neighbours across all class nodes. S_{mean} must be larger than one, since the size of the neighbourhood set S is at least 1 for all nodes if the graph is connected. That is, the complexity of the original version of the ELHNB algorithm depends on both the topology of the class hierarchy (class hierarchies with a higher mean number of neighbours having higher complexity) and the number of classes.

4.3 Experiments

In this section we present the effects of the modification made in the original ELHNB algorithm with respect to predictive performance, training time, and prediction time. To test the algorithm, we use 18 bioinformatics datasets, eight encoding protein functions and ten encoding gene functions.

4.3.1 Datasets

We conducted our experiments in the same 18 datasets made available by Merschmann and Freitas (2013) [1]. These datasets are commonly used in works about hierarchical classification. For the sake of organisation, these datasets are divided in two groups: Group A contains eight protein function datasets, four related to G Protein-Coupled Receptors (GPCRs) and four related to enzymes. GPCRs are transmembrane proteins that are common targets of many medical drugs. Enzymes are large molecules that speed up certain biochemical reactions. The names of the datasets related to enzymes start with EC (Enzyme Commission) and the names of the datasets related to GPCR proteins start with GPCR. The class hierarchies for these datasets are specific to these two types of proteins (GPCRs and Enzymes). The predictor attributes of datasets in Group A include many binary

[1] `http://www.decom.ufop.br/luiz/resources/`

attributes that represent the existence (or not) of a particular protein signature (motif) in a protein amino acid sequence, and two continuous attributes: the amino acid sequence length and the molecular weight. The second part of the dataset name represents the type of motif that was used to create the dataset (Interpro, FigerPrints, Prosite and Pfam). The creation of these datasets is described in (Holden and Freitas 2008).

Group B contains different types of datasets related to the Yeast genome, namely attributes representing: secondary structure, phenotype, homology, sequence statistics, and gene expression. The class labels to be predicted were extracted from the FunCat taxonomy of protein functions. The creation of these datasets is described in (Clare and King 2003). Table 4.1 presents the main characteristics of the datasets used in this work, where "#" means "number of". In the last column of this table, the values separated by "/" are the numbers of classes for the first, second, third, and fourth class levels, respectively.

Figures 4.2, 4.3, and 4.4, display part of the used class hierarchies.

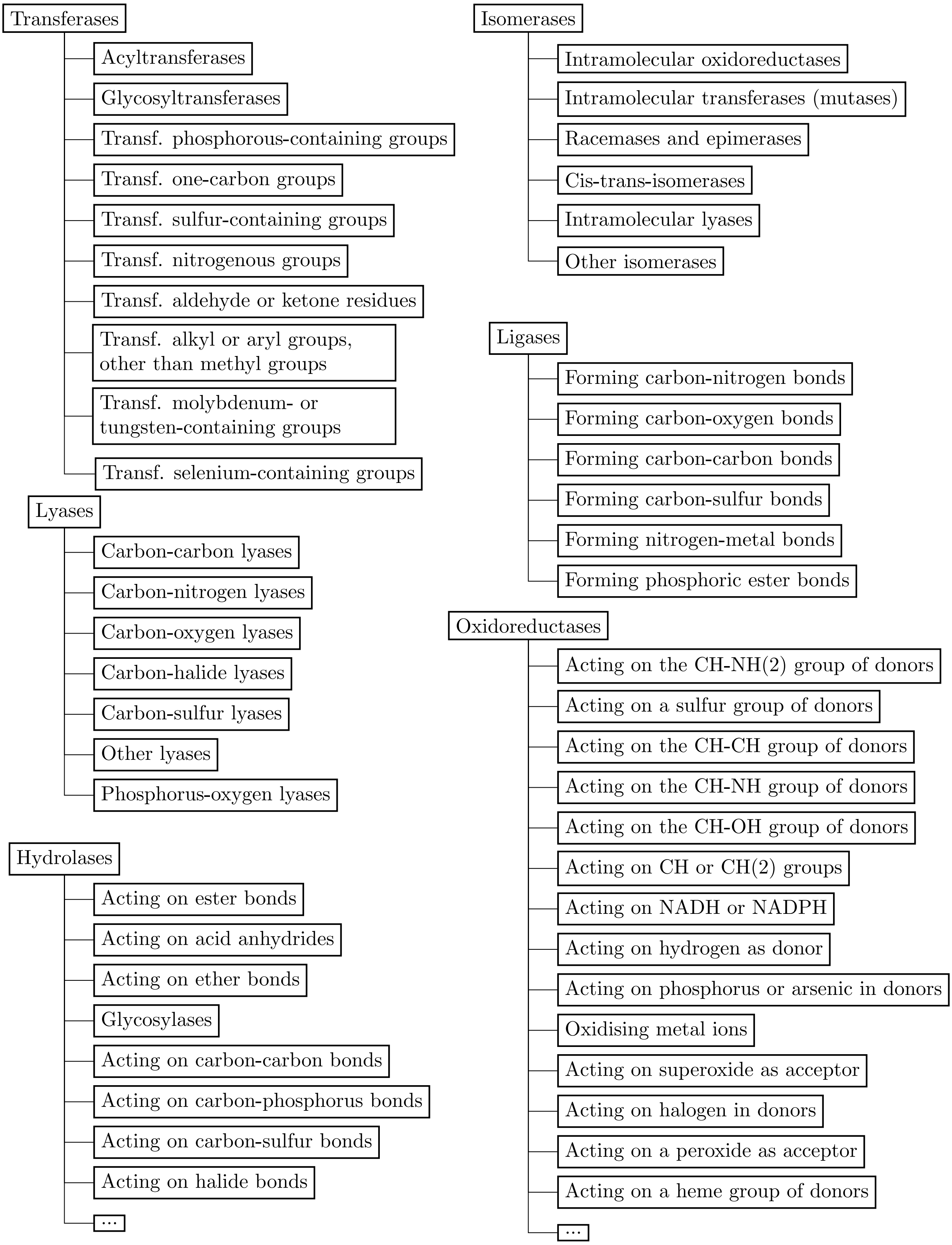

Transferases
Acyltransferases
Glycosyltransferases
Transf. phosphorous-containing groups
Transf. one-carbon groups
Transf. sulfur-containing groups
Transf. nitrogenous groups
Transf. aldehyde or ketone residues
Transf. alkyl or aryl groups, other than methyl groups
Transf. molybdenum- or tungsten-containing groups
Transf. selenium-containing groups
Lyases
Carbon-carbon lyases
Carbon-nitrogen lyases
Carbon-oxygen lyases
Carbon-halide lyases
Carbon-sulfur lyases
Other lyases
Phosphorus-oxygen lyases
Hydrolases
Acting on ester bonds
Acting on acid anhydrides
Acting on ether bonds
Glycosylases
Acting on carbon-carbon bonds
Acting on carbon-phosphorus bonds
Acting on carbon-sulfur bonds
Acting on halide bonds
...
Isomerases
Intramolecular oxidoreductases
Intramolecular transferases (mutases)
Racemases and epimerases
Cis-trans-isomerases
Intramolecular lyases
Other isomerases
Ligases
Forming carbon-nitrogen bonds
Forming carbon-oxygen bonds
Forming carbon-carbon bonds
Forming carbon-sulfur bonds
Forming nitrogen-metal bonds
Forming phosphoric ester bonds
Oxidoreductases
Acting on the CH-NH(2) group of donors
Acting on a sulfur group of donors
Acting on the CH-CH group of donors
Acting on the CH-NH group of donors
Acting on the CH-OH group of donors
Acting on CH or CH(2) groups
Acting on NADH or NADPH
Acting on hydrogen as donor
Acting on phosphorus or arsenic in donors
Oxidising metal ions
Acting on superoxide as acceptor
Acting on halogen in donors
Acting on a peroxide as acceptor
Acting on a heme group of donors
...

Figure 4.2: The Enzyme Commission (EC) hierarchy displaying part of the first and second class levels. The root node, parent of classes Transferases, Lyases, Isomerases, Ligases, Hydrolases and Oxidoreductases, is suppressed for simplicity. Ellipses represent suppressed terms of larger sub-hierarchies (to save space).

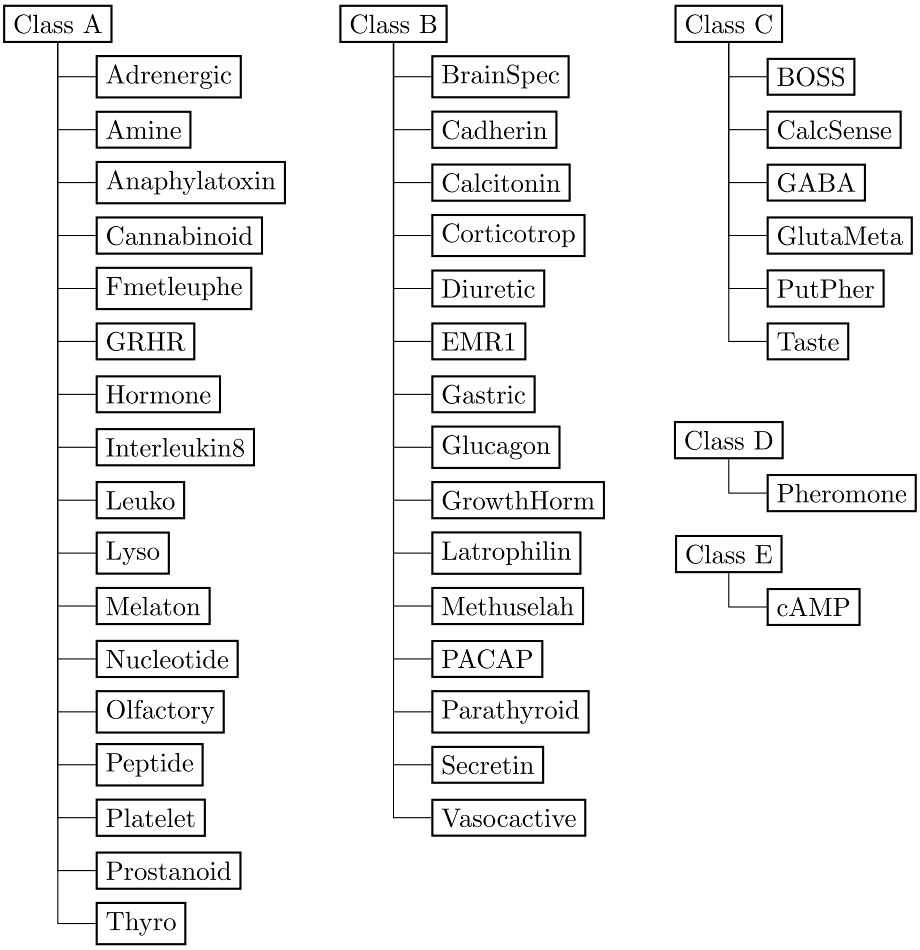


Figure 4.3: Complete first and second class levels of the GPCR hierarchy (the root node, parent of Classes A-E, is suppressed for simplicity).

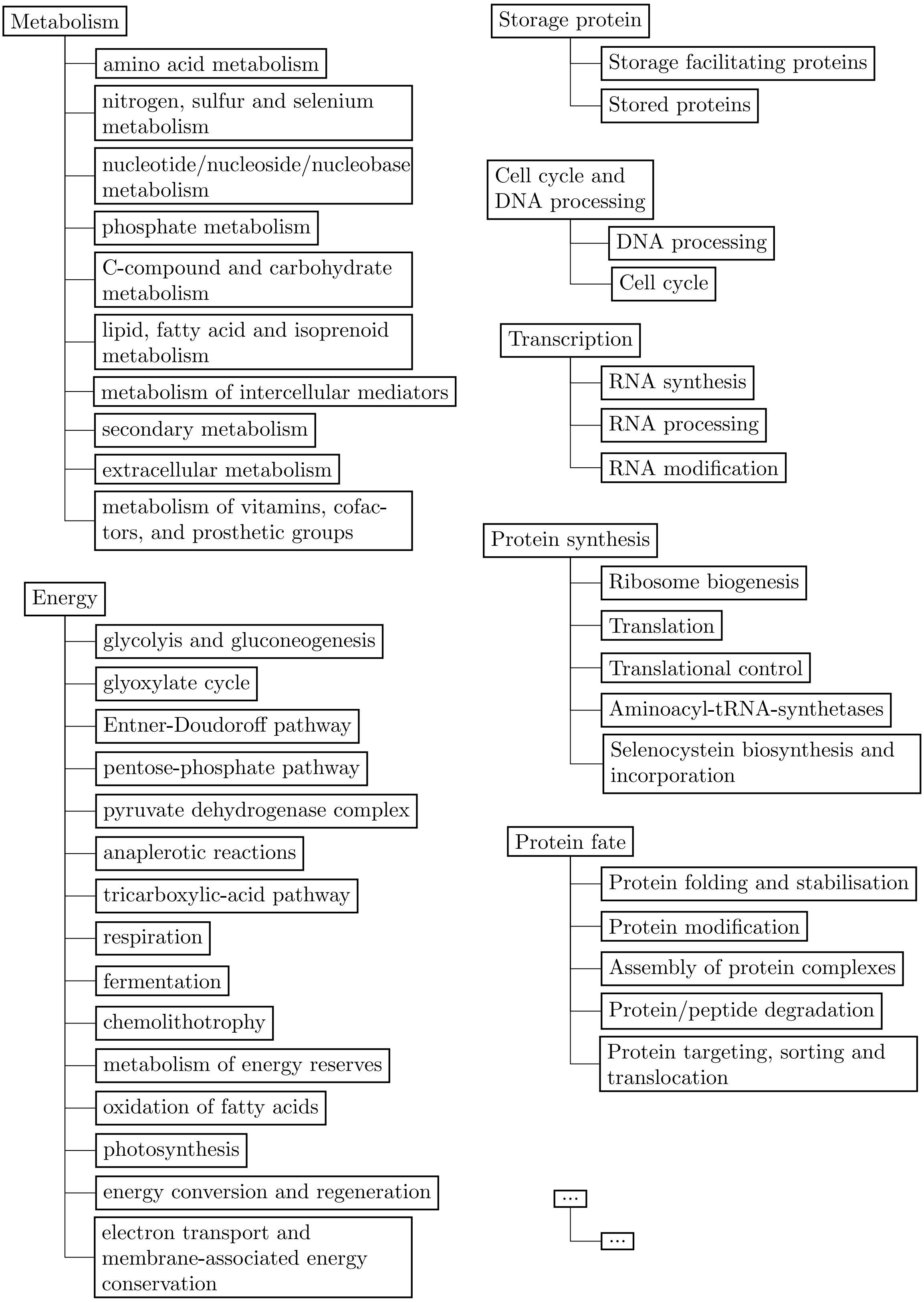

Metabolism
amino acid metabolism
nitrogen, sulfur and selenium metabolism
nucleotide/nucleoside/nucleobase metabolism
phosphate metabolism
C-compound and carbohydrate metabolism
lipid, fatty acid and isoprenoid metabolism
metabolism of intercellular mediators
secondary metabolism
extracellular metabolism
metabolism of vitamins, cofac-tors, and prosthetic groups
Energy
glycolyis and gluconeogenesis
glyoxylate cycle
Entner-Doudoroff pathway
pentose-phosphate pathway
pyruvate dehydrogenase complex
anaplerotic reactions
tricarboxylic-acid pathway
respiration
fermentation
chemolithotrophy
metabolism of energy reserves
oxidation of fatty acids
photosynthesis
energy conversion and regeneration
electron transport and membrane-associated energy conservation
Storage protein
Storage facilitating proteins
Stored proteins
Cell cycle and DNA processing
DNA processing
Cell cycle
Transcription
RNA synthesis
RNA processing
RNA modification
Protein synthesis
Ribosome biogenesis
Translation
Translational control
Aminoacyl-tRNA-synthetases
Selenocystein biosynthesis and incorporation
Protein fate
Protein folding and stabilisation
Protein modification
Assembly of protein complexes
Protein/peptide degradation
Protein targeting, sorting and translocation
...
...

Figure 4.4: The Functional Category (FunCat) hierarchy displaying some of the first level classes and their children. The root node, parent of classes Metabolism, Energy, Storage protein, Cell cycle and DNA processing, Transcription, Protein synthesis and Protein fate, is suppressed for simplicity. Ellipses represent suppressed first and second level terms (to save space).

4.3.2 Predictive Performance

In this section we measure the impact of the proposed modification to the ELHNB algorithm in regards to predictive performance. We use the well-known hierarchical F-Measure to measure predictive performance. The hierarchical F-Measure (hF) is defined as (Kiritchenko, Matwin and Famili 2005)

$$hF \equiv \frac{2 * hP * hR}{hP + hR},$$

where hP is the hierarchical precision and hR is the hierarchical recall, defined as

$$hP \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |P_j|} \quad \text{and} \quad hR \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |T_j|},$$

where P_j is the set of predicted class labels of the j-th instance and T_j is the set of true class labels of the j-th instance.

Table 4.2 presents the hierarchical F-Measure of the modified and original algorithm in the 18 datasets. As observed in this table, the proposed M-ELHNB obtained a higher hF value than the original in 11 out of the 18 datasets, whilst ELHNB obtained a higher hF value in the other 7 datasets. By analysing the results we concluded that in general, for Group A, the M-ELHNB algorithm performed better than the ELHNB in the EC datasets, where the overall size of the datasets is larger. The EC datasets have more features and more class labels to be predicted (more leaf class labels); also they have, on average, more attributes (234.25 vs. 722.75). We conclude that the size of the dataset may negatively affect the original algorithm more than our modified version.

When analysing the results for Group B, we observed that the number of attributes of the datasets is closely related to the performance of the algorithms: the M-ELHNB algorithm performed better when the number of attributes is smaller. More precisely, M-ELHNB outperformed ELHNB in all 7 datasets where the number of attributes in smaller than or equal to 80; whilst ELHNB outperformed M-ELHNB in the other 3 datasets, which have many more attributes (173, 478, and 551 attributes). This may be due to the fact that approximating more Z values using Z_{pseudo} (when there are more attributes to model) negatively affects the predictive performance of M-ELHNB more than approximating relatively fewer Z values (when there are fewer attributes to model).

Table 4.1: Main characteristics of the datasets

Group	Datasets	# Attributes	# Instances	# Classes per level Level
A	GPCR-Interpro	450	6935	12/54/82/50
	GPCR-Pfam	75	6524	12/52/79/49
	GPCR-Prints	283	4880	8/46/76/49
	GPCR-Prosite	129	5728	9/50/79/49
	EC-Interpro	1216	11101	6/41/96/187
	EC-Pfam	708	11057	6/41/96/190
	EC-Prosite	585	11328	6/42/89/187
	EC-Prints	382	11048	6/45/92/208
B	CellCycle	77	2486	16/47/69/32/8
	Church	27	2499	16/49/67/34/6
	Derisi	63	2497	16/48/70/31/7
	Eisen	79	1641	16/43/55/23/2
	Expr	551	2554	16/49/68/28/5
	Gasch1	173	2595	16/48/71/32/7
	Gasch2	52	2631	17/49/68/34/6
	Phenotype	69	1023	15/43/40/15/1
	Sequence	478	2689	17/48/65/29/5
	SPO	80	2463	16/48/68/31/8

Note that the Group B's datasets have relatively small variations in the number of instances and the number of classes per level by comparison with the Group A's datasets. This reinforces the conclusion that, out of the dataset characteristics reported in Table 4.1, the number of attributes is the main characteristic associated with the performance difference between M-ELHNB and ELHNB across Group B's datasets.

Note that the results of the ELHNB and M-ELHNB algorithms seem to be somewhat contradictory between dataset groups A and B. In group A, the ELHNB algorithm seems to outperform algorithm M-ELHNB in smaller datasets, while in group B the opposite behaviour can be observed. This behaviour is intriguing and, unfortunately, we could not find a convincing explanation for this, and further investigations are left for future work.

To test whether the hF values of the modified algorithm (M-ELHNB) are statistically equivalent to the hF values of the standard ELHNB, considering the combined results of all datasets, we used the two-sided Wilcoxon Signed Rank test (Demsar 2006). According to this statistical test, it is not possible to reject the null hypothesis that the algorithms are equivalent (p-value of 0.673 for $\alpha = 0.05$).

Table 4.2: Comparison between the Standard (ELHNB) and the Modified Algorithm (M-ELHNB) in regards to the hierarchical F-Measure using 10-fold cross validation. Numbers in brackets are the standard errors. Numbers in bold face indicate the best performing algorithm for each dataset.

Group	Dataset	ELHNB hF	M-ELHNB hF
A	GPCR-Pfam	**0.6087** (0.0042)	0.5926 (0.0037)
	GPCR-Prosite	**0.5893** (0.0068)	0.5597 (0.0070)
	GPCR-Prints	**0.7689** (0.0048)	0.7567 (0.0054)
	GPCR-Interpro	**0.7693** (0.0021)	0.7573 (0.0042)
	EC-Prints	0.9360 (0.0022)	**0.9451** (0.0016)
	EC-Prosite	0.9490 (0.0022)	**0.9686** (0.0010)
	EC-Pfam	0.9604 (0.0016)	**0.9748** (0.0012)
	EC-Interpro	0.9605 (0.0021)	**0.9772** (0.0013)
B	CellCyle	0.0898 (0.0084)	**0.1382** (0.0064)
	Church	0.0881 (0.0049)	**0.0961** (0.0045)
	Derisi	0.0751 (0.0045)	**0.0825** (0.0033)
	Eisen	0.0489 (0.0023)	**0.1482** (0.0038)
	Expr	**0.0536** (0.0053)	0.0347 (0.0025)
	Gasch1	**0.0674** (0.0059)	0.0178 (0.0027)
	Gash2	0.1153 (0.0061)	**0.1307** (0.0056)
	Phenotype	0.0761 (0.0062)	**0.0809** (0.0072)
	Sequence	**0.0434** (0.0025)	0.0274 (0.0021)
	SPO	0.0741 (0.0043)	**0.1091** (0.0060)

4.3.3 Running time

To test the running time of the two algorithms we measured the total running times of the training and prediction phases of all 10 steps of the 10-fold cross-validation. All algorithms were executed in a cluster computer with 12 Xeon E5520 processors (4 cores each) with 12 GB of RAM memory, running Ubuntu 12.04. We used the Oracle Grid Engine to distribute the jobs. Both algorithms were implemented by the author of this thesis in the Python programming language. Note that our pure Python code is probably not the most efficient implementation of our algorithms, since Python implementations are notoriously slower than implementations in other, lower-level, programming languages.

Figures 4.5 to 4.7 present the running times of the algorithms in the 18 datasets. We divided the datasets into three groups, according to their overall running time. Group I contains the smallest running times and Group III the largest. It is clear from the figures that the modified algorithm has a better running time in both the training and in the prediction phases, as expected. The difference in running times is specially significant in Group III: e.g., in the dataset “EC-Pfam” the ELHNB algorithm took 88.2 hours to run its training and prediction phases, while the modified algorithm took only 37.2 hours. A difference of more than 2 days, and a reduction of about 58% in the time taken by ELHNB. In the dataset “EC-Interpro”, the training running time of the ELHNB algorithm was 58.1 hours, 26.9 hours more than the training running time of the M-ELHNB algorithm, a reduction of 31.7%. The differences in testing running time are even greater: in the dataset “EC-Interpro”, the ELHNB algorithm took 91.1 hours, while the M-ELHNB 33.9 hours, a reduction of 62.8%.

4.4 Conclusions

In this chapter we presented a modified version of the ELHNB algorithm specialised for hierarchical classification problems where each instance is assigned a single path from the root node to a leaf node in the class hierarchy – called single path, mandatory leaf class prediction problems. In experiments with 18 datasets from the area of bioinformatics, involving the prediction of gene or protein functions, we showed that our modified algorithm is statistically equivalent to the standard one in terms of predictive performance, but significantly faster.

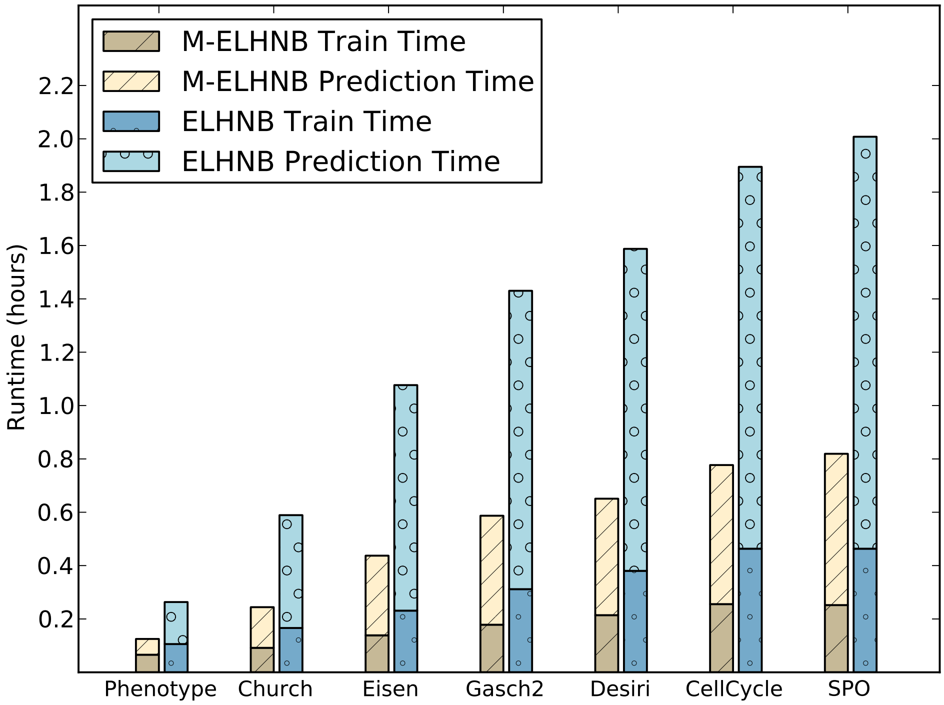


Figure 4.5: Running time of the classifications algorithms, Group I

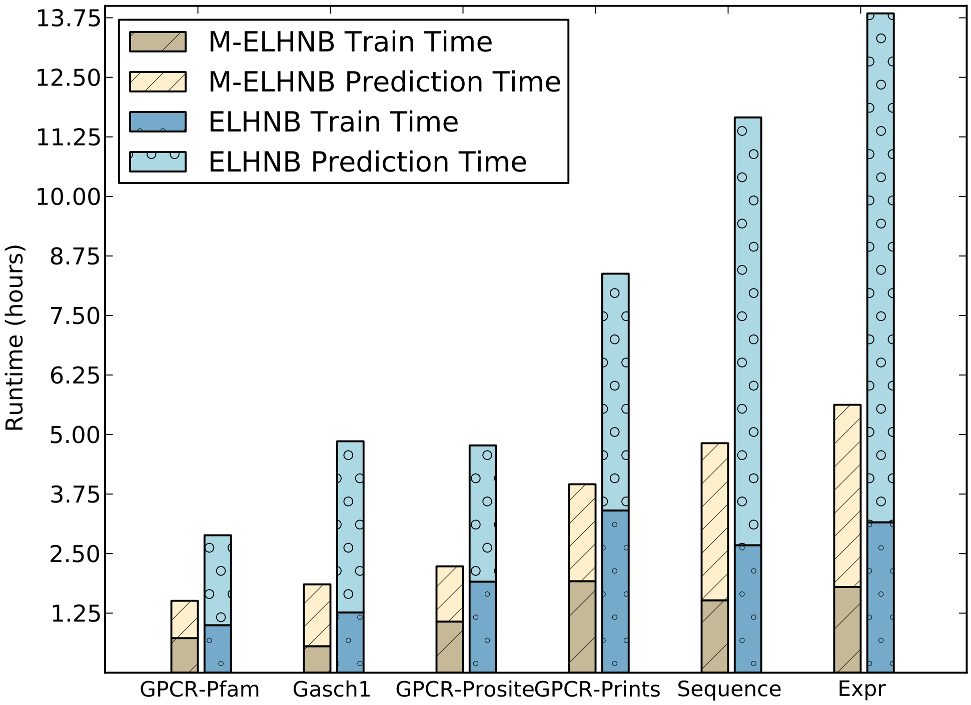


Figure 4.6: Running time of the classifications algorithms, Group II

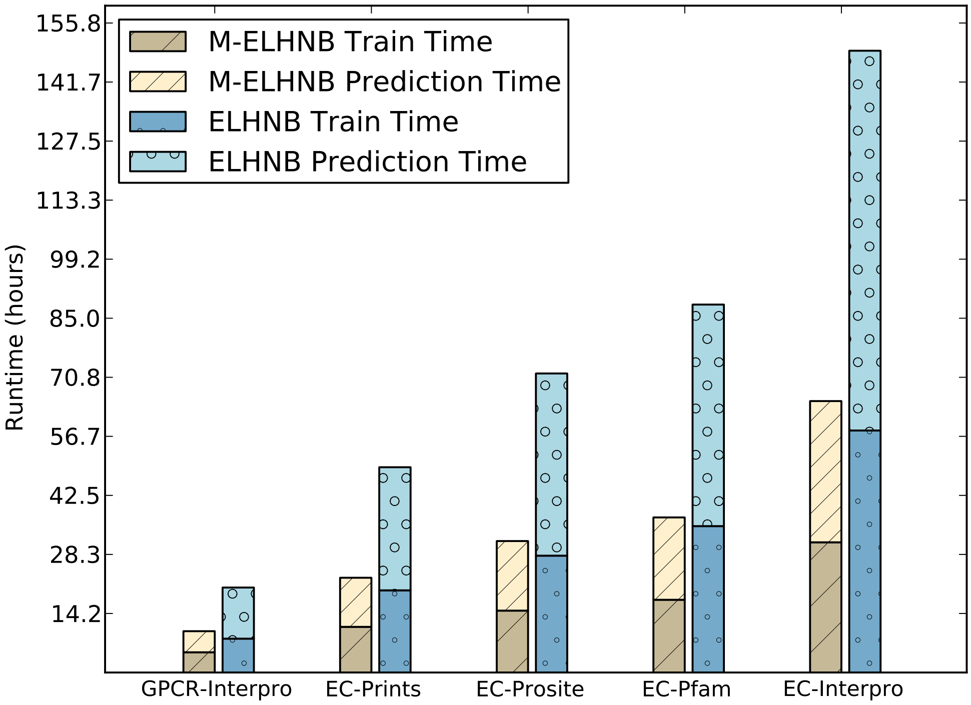


Figure 4.7: Running time of the classifications algorithms, Group III

Chapter 5

New Algorithms for Hierarchical Classification

This chapter presents the new hierarchical classification algorithms we have proposed. Sections 5.2 and 5.3 define our first and second (respectively) Hierarchical Dependence Network algorithms, namely "Hierarchical Dependence Network (HDN)" and the "hybrid between the HDN and Predictive Clustering Tree (HDN-PCT) algorithms", both of them first proposed in (Fabris and Freitas 2014b). Section 5.4 presents our third hierarchical classifier, named "HDN based on finding non-Hierarchically related Predictive Classes (HDN-nHPC)", first presented in (Fabris and Freitas 2015). Section 5.5 defines the hybrid "Predictive Clustering Tree/Local Hierarchical Classification" (PCT-LHC) algorithm, proposed by us in (Fabris and Freitas 2014b).

5.1 Introduction

Dependence Networks (DNs) are a type of Probabilistic Graphical Model (PGM) that has not received as much attention in the data mining field as other types of PGMs such as Bayesian Networks, and its variants, and Markov networks. Some works explore the use of DNs in binary and *flat* multi-label classification (Gámez et al. 2008; Guo and Gu 2011); however, as far as we know, besides our work, there are no attempts of using DNs in hierarchical classification.

Like in Bayesian Networks (BNs), variables in DNs are represented as nodes in a graph. However, DNs differ from BNs by the fact that, in a DN, edges exist between nodes if and only if they are in each other's Markov blankets, and the graph may contain cycles (Heckerman et al. 2001). The Markov blanket of a random variable r is the smallest set containing the variables that make r independent from all other variables. In other words, given a node in a DN, if we know the actual values of the nodes in its Markov blanket, our beliefs about that node will not change if we know the value of some other node outside the Markov blanket.

In BNs, the Markov blanket of a node is the set containing its children, parents and parents of the children, that is, the Markov blanket of a node is not explicitly represented by the graphical structure (Heckerman et al. 2001). Instead, edges represent (potentially) causal relationships between variables. For this reason, some relationships are not explicitly represented in the graph structure of BNs; therefore people must be trained to interpret the graphical representation correctly (Heckerman et al. 2001).

Figure 5.1 exemplifies the representation of the same relationships in a BN (at the top of the figure) and in a DN (at the bottom of the figure). Recall that in DNs, the neighbours of a node represent the Markov blanket of that node. That is, the minimal set of nodes that makes the variable independent of all other variables. For instance, in Figure 5.1, if we know the condition of the tires and the maximum speed of the car, knowing the tire brand would not change our beliefs about the car weight. The intuition behind this is that the influence that the tire brand has on the car weight is indirect, through the condition of the tires (heavier cars tend to have worse tires), and once we know the actual condition of the tires, knowing the tires' brand adds no extra information to our beliefs.

Note that an untrained person would look at the BN at the top and wonder why there is no edge from "Car Weight" to "Tire Condition", since knowing the car weight clearly affects our beliefs about the condition of the tires. This confusion is avoided in DNs by explicitly representing the relationships between nodes that are directly correlated, as shown at the bottom of Figure 5.1.

DNs for classification can be thought of as a collection of probability functions specialised in predicting whether or not an instance belongs to a class given its predictive attributes and the predictions of the classes in its Markov blanket. More formally, given:

- the set of binary random variables $\mathbf{C} = \{C_1, \ldots, C_N\}$ representing the classes,

- their possible values c_i and $\tilde{c}_i$, where $C_i = c_i$ is the event that the i-th class is assigned to the instance and $C_i = \tilde{c}_i$ the event that the i-th class not is

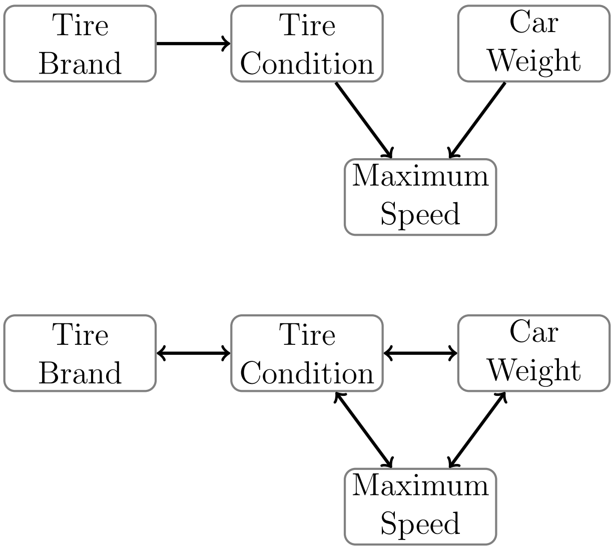


Figure 5.1: Representing the same probabilistic relations with Bayesian Networks (top) and Dependence Networks (bottom).

assigned to the instance,

- the vector $\mathbf{x} \in \mathbb{R}^M$ representing the predictive features of the instance, where M is the number of features,
- a directed graph $G = (V, E)$, where V is the set of vertices (nodes) and E is the set of edges,
- the set of parents of node v_i ($v_i \in V$), denoted pa_i, representing the Markov blanket of C_i;

a DN is a graph, with a probability distribution $P(C_i = c_i|pa_i, \mathbf{x})$ associated with each node $v_i \in V$, where E is the set of directed edges representing dependencies between variables. From now on, for simplicity, we shorthand the probabilities in the form $P(C_i = c_i|\cdot)$ and $P(\cdot|C_i = c_i, \cdot)$ to $P(C_i|\cdot)$ and $P(\cdot|C_i, \cdot)$, respectively.

In the hierarchical classification task, each node in a DN encodes probability functions in the form $P(C_i|\mathbf{x}, \mathbf{C_{-i}})$, where $\mathbf{C_{-i}}$ is the vector of predictions of the classes in the Markov blanket of C_i.

There are two main challenges when using DNs; the first is creating the DN itself. This involves: 1) determining the structure of the network, or in other words, estimating the Markov blanket of each random variable; and 2) modeling the distribution of each random variable given its Markov blanket and the feature vector (the probability function $P(C_i|\mathbf{x}, \mathbf{C_{-i}})$). Recall that, as discussed in

Chapter 2, one can either make strong simplifications about the data distribution or use some methodology to empirically find the correlations and distributions of the random variables being modeled. The second challenge is, once the network is fully parameterised, how to query the network to make predictions, since exact inference in DNs is an NP-Hard problem (Guo and Gu 2011).

Specifically for hierarchical classification, to solve the first problem, we can use traditional *flat* classifiers that can be trained independently to determine both the structure of the network (e.g. by using feature selection in a pre-processing step or a classifier that selects a subset of features) and the estimation of $P(C_i|\mathbf{x}, \mathbf{C_{-i}})$ for each class C_i. We call those flat classifiers "base classifiers". In other words, we rely on a feature selection method to find random variables (features and classes) that are correlated and, thus, should be in each other's Markov blankets.

To solve the second problem, if we relax the requirements for exact inference, there is a simple, yet effective, algorithm, *viz.* Gibbs sampling, that is capable of producing approximate inference in reasonable running times. This is particularly important for hierarchical classification because of the large number (often thousands) of classes in real-world problems, such as gene function prediction.

In the next sections we describe the three variations of Hierarchical Dependence Network algorithms that we have proposed.

5.2 Hierarchical Dependence Network Using Hierarchically Related Predictive Classes

For this variation of Hierarchical Dependence Network (HDN) for hierarchical classification, we assume that the set of class variables containing the siblings and the parents of the children of the class C_i in the class hierarchy is a good candidate for the $\mathbf{C_{-i}}$ part of the Markov blanket of C_i, because they encode important relationships about the classes that are not deterministic. Initial experimentation also included the set of class variables consisting of the parents and the children of the class C_i as part of the Markov blanket of C_i; however, it was observed that the classifiers did not cope well with such deterministic parent-child relationships, defined by the "Is-A" class hierarchy. That is, if an instance is labeled with any of the child classes of C_i, by definition of the "Is-A" hierarchy, the instance must be labeled with class C_i as well. Similarly, if the instance is not labeled with any of the parent classes of C_i, by definition, it cannot have the class label C_i. It turned out that these deterministic features misled the HDN algorithm, biasing them to produce over-simplified class models that underfit their predictions to these deterministic features.

This kind of underfitting occurred because, although deterministic parent-child relationships are very useful for achieving a good classification accuracy in the training set (where both the parent class(es) and the child class(es) of a given current class C_i are known); such parent-child relationships are less useful in the test set, where the parent class(es) and child class(es) of C_i are unknown and have to be estimated by Gibbs sampling (described later).

The use of the siblings and parents of the children of class C_i as extended features exploits background knowledge (the structure of the class hierarchy) defined by expert biologists in the case of the datasets used in this work, and avoids the computationally expensive search for the C_{-i} part of the Markov blanket of each node. Note that the conditional probability $P(C_i|\mathbf{x}, \mathbf{C_{-i}})$ assumes that every class in $\mathbf{C_{-i}}$ and every predictive feature in $\mathbf{x}$ is in the Markov blanket of C_i. However, there may be noisy predictive features or classes in C_{-i} that are not good predictors for C_i. This may misguide a classifier for C_i and clutter the DN. For this reason, we reduce the size of the Markov blanket of C_i by applying a feature selection filter on $\mathbf{x}$ and $\mathbf{C_{-i}}$ (treating them as a single, extended predictive vector), generating a new filtered predictive feature vector $\mathbf{x}^+$. This is detailed next.

Training Phase

The training phase of our HDN for hierarchical classification is similar to the training phase of DN for binary classification, with the exception that we extend the predictive feature vector $\mathbf{x}$ of each instance to contain the class labels of the current class node's candidate Markov blanket. That is, for each class label C_i, we first use a simple feature selection method to produce the filtered extended predictive feature vector $\mathbf{x}^+$, as mentioned earlier. We use the statistical F-test (Lomax and Hahs-Vaughn 2013) to select relevant features for each class C_i.

More precisely, the univariate F-test for feature selection is a method that ranks the features, for each class label i, according to their F-value, defined as:

$$F_{i,k} = \frac{SSBETW_{i,k}}{SSWITH_{i,k}}, \tag{5.1}$$

where $SSBETW_{i,k}$ is the between-group sum of mean squares and $SSWITH_{i,k}$ is the within-group sum of mean squares for feature k, according to the groups defined by the i-th binary class label. $SSBETW_{i,k}$ and $SSWITH_{i,k}$ are defined as:

$$SSBETW_{i,k} = \sum_{c' \in \{c_i, \tilde{c_i}\}} N_{c'}(Avg_{c',k} - Avg_k)^2, \text{and} \tag{5.2}$$

$$SSWITH_{i,k} = \sum_{c' \in \{c_i, \tilde{c_i}\}} \frac{(N_{c'} - 1)s^2_{c',k}}{J - 1}. \tag{5.3}$$

Where $N_{c'}$ is the number of training instances annotated with class value c', J is the total number of training instances, $Avg_{c',k}$ is the mean value of the k-th feature considering only training instances with class value c', Avg_k is the mean value of the k-th feature across all training instances, and $s^2_{c',k}$ is the variance of the k-th feature considering only training instances with class value c'.

The higher the F-value is, the greater is the difference in feature k's values between the groups. In our case, when dealing with local binary classifiers, we always have two groups, defined by the class label of the instances. Hence, the larger the F-value, the more relevant feature k is for class discrimination.

We have chosen to use the F-test in our algorithm due to its statistical soundness and its computational efficiency, which is important when dealing with thousands of classes and features, like the classes and features of the datasets used in our experiments. Note, however, that there are other univariate feature selection methods that are also computationally efficient (Bramer 2013), and other methods could be used in future work. In preliminary experiments we have also tested the multivariate Correlation Feature Selection (CFS) algorithm (Hall 1999), however, because of its higher time complexity in comparison with the F-test, its application in our datasets was too time consuming.

We select the top *n_feats* features in the ranking of features produced by the F-test for each class label i, where *n_feats* is a parameter of the F-test feature selection method. Next, we train a binary classifier $\mathcal{F}_i$ with the filtered feature vector $\mathbf{x}^+$, which contains both the selected features and the selected actual class labels of the nodes in the Markov blanket of class C_i, considering the given class taxonomy. We expect that the classifiers use the extended information to learn dependencies between class labels and predictive features.

In this work we have experimented with classification models capable of outputting probabilistic classification for all class labels of the instances, namely, Gaussian Naive Bayes (GNB), Naive Bayes (NB), SVM (with RBF kernel) and C4.5 decision tree. We have tried both NB and GNB classifiers because the NB classifier requires a discretisation of the data set (as many datasets have continuous features) that may affect its predictive performance, and GNB avoids the need for such discretisation. On the other hand, GNB assumes that the continuous features are normally distributed, which may not be true for many features. Hence, it is worth trying both these versions of Naive Bayes. The popular SVM classification algorithm is regarded as a very powerful classifier, with high predictive accuracy. The C4.5 decision tree algorithm also has its advantages: it has a built-in feature selection procedure, and it generates interpretable models. We consider that these four algorithm (NB, GNB, SVM, and C4.5) are diverse, having different assumptions and principles, and are in general popular and powerful, having being tested in several data mining tasks with success.

After testing these four algorithms in our preliminary experiments, we have concluded that the SVM is in general the best for our purposes, having higher predictive performance. In addition, like the other classification algorithm used in our preliminary experiments, SVM also has an efficient prediction phase (Claesen et al. 2014). It is particularly important in this work that the base classification algorithms have an efficient prediction phase, as they will be queried several times during the Gibbs sampling process (explained in the next section). For these reasons, from now on, we use the aforementioned SVM algorithm to build the functions $\mathcal{F}_i$.

For all versions of HDN algorithms that we propose, when training the algorithm for each class node C_i, we use as positive classes all instances that have the class label C_i in any of their classification paths (all paths in the class taxonomy graph from the *root* node to the most specific classes of the instances) and as negative classes all the other instances. Furthermore, we only train a classification algorithm for a particular class node if there are at least *min_inst* instances in the least represented class (i.e., positive or negative class), in order to mitigate the problem of overfitting – where *min_inst* is a parameter of the HDN algorithm. If a classifier is not trained for a class node, the classification model outputs the *a priori* class probability distribution for that class node, regardless of the filtered extended predictive feature vector $\mathbf{x}^+$. The *a priori* distribution is given by the relative frequency of each class label in the training set for that class node.

Gibbs Sampling for Approximate Inference

Once we have trained our binary classifiers to estimate the probabilities $P(C_i|\mathbf{x}, \mathbf{C_{-i}})$ – which are estimated by using the filtered extended feature vector $\mathbf{x}^+$, i.e. computing $P(C_i|\mathbf{x}^+)$ – we already have a fully parameterised HDN where each class node has edges connecting it to the selected features and the selected predictive classes. The next step is to use our HDN algorithm to predict a class vector $\mathbf{C}$ for a new instance given the predictive features $\mathbf{x}^+$. In other words, we wish to calculate the maximum *a posteriori* (MAP) probability for our predictive feature vector: $\mathbf{C}^* = \arg\max_{\mathbf{C}} P(\mathbf{C}|\mathbf{x}^+)$.

Because solving this problem exactly is NP-hard (Guo and Gu 2011), we use the Gibbs sampling algorithm adapted to our hierarchical classification problem (presented in Algorithm 5.1) to do inference in our DN. The Gibbs sampling algorithm (Geman and Geman 1984) is a Metropolis-Hastings algorithm that is very suitable for inference in DNs, due to its low run-time complexity compared to the exact algorithm, and has a simple and efficient implementation (Guo and Gu 2011). After a certain number of burn-in iterations (a user-defined parameter), this algorithm tends to converge to a stationary distribution that approximates the underlying probability distribution.

The algorithm begins by randomly initialising $\mathbf{C_{pred}}$, the class vector of the instance whose classes will be predicted, using the *a priori* class probabilities computed from the training set (line 2). Next, the algorithm visits the nodes of the graph representing the class hierarchy using some node ordering (line 4). We have explored three ordering possibilities: bottom-up (start by visiting the leaves of the hierarchy, recursing to its parents, in a breadth-first manner), top-down (start by visiting the *root* node and recursing to its children in a breadth-first manner) and random order. The random ordering produced results with better predictive accuracy in our preliminary experiments (using a single dataset), so we will use this ordering from now on.

In line 5, the function ExtendedFeatures(C_i) returns the class labels that are candidate to be in the Markov blanket of C_i (i.e., the siblings of C_i and the parents of the children of C_i in the class hierarchy). In line 6, the function ApplyFeatSel$_i(\mathbf{x}, \mathbf{C_{-i}})$ filters the predictive features and candidate class labels using the previously mentioned statistical F-test for the i-th class, returning the filtered extended vector $\mathbf{x}^+$. Next, the algorithm retrieves the probabilistic classification of the current node C_i given the current class labels of its Markov blanket

Algorithm 5.1 Hierarchical classification with Gibbs sampling algorithm.

```
1:  procedure MODIFIED GIBBS SAMPLING
    Inputs: Instance $\mathbf{x}$, total number of iterations $it$, number of burn-in
    iterations $burn$.
2:      Probabilistically initialise the predicted class label vector of $\mathbf{x}$ ($\mathbf{C_{pred}}$),
        using uniform distributions (parametrised by the a priori empirical class
        distributions).
3:      for $k \in \{1..it\}$ do
4:          for all $C_i \in$ the Class Taxonomy do
5:              $\mathbf{C_{-i}} \leftarrow \text{ExtendedFeatures}(C_i)$
6:              $\mathbf{x}^{+} \leftarrow \text{ApplyFeatSel}_i(\mathbf{x}, \mathbf{C_{-i}})$
7:              $P(C_i|\mathbf{C_{-i}}, \mathbf{x}) \leftarrow \mathcal{F}_i(\mathbf{x}^{+})$
8:              $u \leftarrow$ Random value drawn from a uniform distribution in $[0, 1]$.
9:              if $u < P(C_i|\mathbf{C_{-i}}, \mathbf{x})$ then $C_{pred,i} \leftarrow c_i$ else $C_{pred,i} \leftarrow \tilde{c}_i$ end if
10:             if $k > burn$ then
11:                 $P(C_i|\mathbf{x}) \leftarrow \frac{P(C_i|\mathbf{x}) \times (k - burn - 1) + P(C_i|\mathbf{C_{-i}}, \mathbf{x})}{k - burn}$
12:             end if
13:         end for
14:     end for
15:     return the marginal probabilities $P(C_i|\mathbf{x})$
16: end procedure
```

and its predictive feature vector (line 7). Finally, the algorithm employs a stochastic rule to update the current classification of the input instance (line 9). There are several ways to estimate the MAP of the predictive classes; we have used the typical way of computing the marginal probability for each class variable and use them for making the final prediction (Guo and Gu 2011).

Our Gibbs sampling algorithm was slightly modified to return the mean probability associated with each class (line 11). The more common approach would be to return the most common label for each class (Sen et al. 2008) (in our case the positive or negative label). We employ an iterative approach to calculate the mean class probabilities over the iterations after the burn-in phase, so there is no need to store all class-wise probability estimations.

As we will compare our methods to other probabilistic classifiers, we modified the Gibbs sampling algorithm in order to obtain a probabilistic decision for each class at the end of the sampling procedure.

We call the previously described training procedure and inference algorithm as the Hierarchical Dependence Network (HDN) algorithm.

5.3 The Hybrid HDN-PCT Algorithm

Besides investigating the HDN algorithm by itself, we also exploit the power of the PCT (Predictive Clustering Tree) framework. An algorithm in this framework builds a decision tree by finding a predictive feature that splits the set of instances in two clusters, maximising the class distribution similarity within each cluster and the dissimilarity of the class distributions across the two clusters. In order for the split to be accepted, the class distribution of the instances across the two clusters must be statistically different according to the F-test. The algorithm recurses in each cluster that it forms and eventually stops if it finds no statistically significant split or the size of a cluster falls bellow a pre-established threshold.

In the prediction phase, to classify an instance $\mathbf{x}$, a PCT algorithm first identifies the cluster associated with that instance and then assigns, to instance $\mathbf{x}$, classes whose value in the mean probability vector of that cluster is greater than a probability threshold. The threshold is varied when computing a Precision-Recall curve, as explained later.

We shall use the most well-known implementation of PCTs for hierarchical classification, the Clus-HMC algorithm (Vens et al. 2008). Clus-HMC obtained good predictive performance in relation to other PCT algorithms.

We apply our HDN algorithm in each cluster given by a leaf node in the decision tree produced by the Clus-HMC algorithm (that we shall call simply PCT from now on), and we name the combination of both algorithms as the HDN-PCT algorithm. This combination introduces another parameter, min_inst_HDN, the minimum number of instances required in each cluster to train our dependence network. If the number of instances in a cluster is below that minimum, the classification of instances associated with that cluster is performed by the PCT algorithm.

Note that, in principle, a HDN model can be trained to substitute the predictions of the internal nodes of the PCT, not just the leaf nodes. This can be done by inducing a HDN model using the training instances reaching an arbitrary node (Nd), and comparing the predictive performance of the sub-tree spanning from Nd and the HDN model, using this information to decide whether to use the sub-tree or the HDN model during testing. This procedure can even be applied in a greedy fashion (recursively applying this procedure from root to leaves, stopping the recursive process at the first HDN model with better performance than the PCT) or in a exhaustive way (testing all possible sub-tree–HDN-model substitutions). These variations are left for future work.

We adopt the following strategy when using our HDN-PCT algorithm: after the training phase we analyse which HDN classifiers had better overall classification accuracy on the validation set (whose instances have no overlapping with the instances in the test set used to evaluate the classifier's predictive performance) in comparison with clusters formed by the baseline PCT algorithm. If the classification accuracy of the HDN for a PCT cluster is higher than the classification accuracy of PCT for that cluster, we analyse each class C_i in that cluster and calculate the mean loss L_i, defined as:

$$L_i \equiv \frac{\sum_j ((\mathbf{1}_{i,j}) - p_{i,j})^2}{n_i}, \tag{5.4}$$

where $\mathbf{1}_{i,j}$ is an indicator function defined as:

$$\mathbf{1}_{i,j} \equiv \begin{cases} 1 & \text{if instance } j \text{ has } C_i \text{ as a true class,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

$p_{i,j}$ is the estimated probability of the j-th instance belonging to the i-th class and n_i is the number of instances of the i-th class.

Considering this loss function, the predictions of the HDN-PCT algorithm and the PCT algorithm on the validation set are compared to the real classifications. Then, for each class, we discard the HDN classifier with greater loss than the loss achieved by the stand-alone PCT algorithm for that class and use the output of the PCT classifier instead for predicting that class.

Notice that it is not possible to calculate ranking-based quality measures for the individual classifiers (for instance AUROC or AUPRC) in a given cluster formed by the PCT algorithm, because the class probabilities assigned to all instances are always the same, which yields an arbitrary instance ranking with respect to the probability of an instance belonging to a given class. This justifies the use of the loss function defined by Equation (5.4).

5.4 Hierarchical Dependence Network Based on finding non-Hierarchically Related Predictive Classes

In this section we present our Hierarchical Dependence Network algorithm based on finding non-Hierarchically related Predictive Classes (HDN-nHPC), our last proposed HDN algorithm variation.

5.4.1 Estimating the Class Blanket of each Class Variable

The HDN-nHPC algorithm relies on the hypothesis that among the vast number of classes (usually hundreds or thousands) in typical hierarchical classification problems, there are class labels that, when present in an instance, change how the feature vector affects the prediction of some other class label. The idea behind the HDN-nHPC algorithm is to find and exploit such dependencies in a data-driven way, using the class hierarchy to guide this search. Then, for each class variable C_i, the predictions of the class variables related with C_i are used to inform the prediction of C_i using the Gibbs sampling algorithm. Next we describe in more detail how the set of related features are identified.

Let $(C_i \leftharpoonup C_{i'})$ be an ordered pair of class variables such that C_i is related to $C_{i'}$. We call the set containing all such pairs of classes as the set $\mathbf{C}^{\dagger}$.

To build the set $\mathbf{C}^{\dagger}$ and the Markov blanket of each class variable $C_i \in \mathbf{C}^{\dagger}$, we created a method to find pairs of classes $(C_i \leftharpoonup C_{i'})$. These pairs of classes have the property that information about class $C_{i'}$ affects the probability distribution of class C_i. That is, knowing whether or not an instance is annotated with class $C_{i'}$ affects our predictions about C_i (i.e. $P(C_i|\mathbf{x}, C_{i'}) \neq P(C_i|\mathbf{x})$).

To achieve this, first we create a candidate set of predictive class relationships, $\mathbf{C_{cand}}$, containing every pair of classes $(C_i \leftharpoonup C_{i'})$ that are not descendants (or ancestors) of each other. This condition is necessary since if classes are descendants (or ancestors) of each other, we would have deterministic relationships between C_i and $C_{i'}$, that is, if C_i is an ancestor of $C_{i'}$, $C_i = \tilde{c}_i \Rightarrow C_{i'} = \tilde{c}_i$ and $C_{i'} = c_i \Rightarrow C_i = c_i$. Recall that $C_i = \tilde{c}_i$ means that a particular class label $C_{i'}$ is not present in a instance, and $C_i = c_i$ otherwise.

As mentioned earlier, for the HDN algorithm these deterministic relationships have been observed, in our preliminary experiments, to produce underfit classification models, i.e., models that rely too much on the values of classes that are descendants or ancestors of the current class being predicted during training, resulting in over-simplified models. Another condition that must be satisfied for a pair of classes to be in $\mathbf{C_{cand}}$ is that both classes, C_i and $C_{i'}$, take the value c_i (rather than $\tilde{c_i}$) in at least *min_inst_Cand* instances. We specify this condition to avoid considering classes with too few instances, thus inducing unreliable classifiers.

The next step is to induce two SVM models to classify C_i, two models for each pair $(C_i \leftharpoonup C_{i'}) \in \mathbf{C_{cand}}$. The first model, $M^{\tilde{c_i}}_{(i,i')}$, is trained using dataset $D^{\tilde{c_i}}_{(i,i')}$ and the second, $M^{c_i}_{(i,i')}$, using dataset $D^{c_i}_{(i,i')}$, where $D^{\tilde{c_i}}_{(i,i')} \cup D^{c_i}_{(i,i')} \equiv D_{learn}$, and D_{learn} is a random partition of 70% of the training set. Instances are assigned to $D^{\tilde{c_i}}_{(i,i')}$ if they have $C_{i'} = \tilde{c_{i'}}$; and conversely, assigned to dataset $D^{c_i}_{(i,i')}$ if $C_{i'} = c_{i'}$. Again, we use SVM because of its good predictive performance on preliminary experiments.

After training the two SVM models using the values of the class variable $C_{i'}$, we need to test if the resulting pair of models (i.e., using $M^{\tilde{c_i}}_{(i,i')}$ and $M^{c_i}_{(i,i')}$ together) has better predictive performance than a single SVM classifier ($M_{(i,i')}$), induced using dataset D_{learn}, without splitting the data based on $C_{i'}$ values.

To do so, we use the training instances in D_{valid}, the complement of D_{learn} in the training set, as validation instances. Note that the classifiers did not have access to these instances earlier. To classify a new instance using the two classifiers, which we shall call a *classifier pair* from now on, we use the actual value of $C_{i'}$ to decide which classifier to use: if $C_{i'} = \tilde{c_i}$ we use $M^{\tilde{c_i}}_{(i,i')}$, if $C_{i'} = c_i$, we use $M^{c_i}_{(i,i')}$.

Intuitively, we would expect that if two given class variables are unrelated, i.e., the presence or absence of one class label does not change the distribution of another class label given the features ($P(C_i|\mathbf{x}, C_{i'}) = P(C_i|\mathbf{x})$), the classifier pair would have similar predictive performance to a single classifier that uses the whole "learning set" D_{learn}. In fact, due to the reduced individual sizes of datasets $D^{\tilde{c_i}}_{(i,i')}$ and $D^{c_i}_{(i,i')}$, it is likely that the classifier pair ($M^{\tilde{c_i}}_{(i,i')}$, $M^{c_i}_{(i,i')}$) would perform worse on average than the single classifier ($M_{(i,i')}$) trained using the whole learning set D_{learn}. On the other hand, if the presence or absence of one class label affects the distribution of another class label given the predictive features, our approach would yield better classifiers than the single-classifier baseline if the classification models in the pair of classifiers can exploit such differences. Using this rationale, we select the pairs $(C_i \leftharpoonup C_{i'})$ whose classifiers achieved better predictive accuracy than the single classifier $M_{(i,i')}$ on the validation set D_{valid} to construct the set of pairs of predictive classes $\mathbf{C}^{\dagger}$, which is a subset of $\mathbf{C_{cand}}$.

With the set $\mathbf{C}^{\dagger}$, we can finally estimate the Markov blanket of each class label C_i, which is divided in two parts: The first part, $\mathbf{C_i^{FB}}$, is the "Feature Blanket of C_i", a subset of $\mathbf{x}$, the minimal set of predictive features that affect the estimation of C_i. This set is estimated using the feature selection method F-test. The second part, $\mathbf{C_i^{CB}}$, is the "Class Blanket of C_i". This is specified by inducing a directed graph G using the set $\mathbf{C}^{\dagger}$. For each pair $(C_i \leftharpoonup C_{i'}) \in \mathbf{C}^{\dagger}$ we create a directed edge from $C_{i'}$ to C_i, representing the fact that the value of $C_{i'}$ affects the value of C_i. Thus, the vertices that point to C_i comprise the set $\mathbf{C_i^{CB}}$.

The procedures discussed thus far are presented in Algorithm 5.2 using the additional notation: $M^{c_i}_{(i,i')}(\mathbf{x})$, $M^{\tilde{c_i}}_{(i,i')}(\mathbf{x})$ and $M_{(i,i')}(\mathbf{x})$ denote the prediction of the corresponding models given the feature vector $\mathbf{x}$. The function $AUPRC(predictions)$ calculates the Area Under the Precision and Recall Curve for a single class, a measure of the predictive quality of the *predictions*, on the validation set D_{valid}.

5.4.2 Estimating Class-Label Probabilities

Once we have graph G, which represents the dependence network among class labels, we train the classifier pairs for the classes in $\mathbf{C}^{\dagger}$ using the whole training set, D, one classifier pair for each class in $\mathbf{C_i^{CB}}$, for all class variables C_i. The next step is to develop a way to query the classifiers of each class variable C_i and get a single prediction to estimate $P(C_i|\mathbf{C_i^{FB}}, \mathbf{C_i^{CB}})$. As each class variable C_i may have several classifier pairs, one pair for each class in $\mathbf{C_i^{CB}}$, to return a unified prediction, we return the average class probability over the probabilities computed by the classifier pairs. That is, we average over a set of the class probabilities computed by the selected classifier in the classifier pair of each class variable C_i (one prediction for each element of $\mathbf{C_i^{CB}}$) using the predicted values (c_i or $\tilde{c_i}$) of the class variables in $\mathbf{C_i^{CB}}$ to choose which models in the classifier pairs to use. This procedure is presented in Algorithm 5.3.

Recall that during the iterations of the Gibbs sampling algorithm, the values of the class variables in the Markov blanket of class variable C_i ($\mathbf{C_i^{CB}}$) are always well-defined (although changing across iterations). So, we can use the current value of each class variable $C_{i'}$ in the Markov blanket of C_i to select which classifier on the classifier pair to use: If $C_{i'} = c_{i'}$, we use the classifier on the classifier pair that was trained to predict C_i only using instances in D with $C_{i'} = c_{i'}$, conversely, if $C_{i'} = \tilde{c_{i'}}$

Algorithm 5.2 Building the set of predictive class relationships.

```
procedure FIND THE SET OF PREDICTIVE CLASS RELATIONSHIPS $\mathbf{C}^{\dagger}$ AND THE GRAPH INDUCED BY $\mathbf{C}^{\dagger}$.
Inputs: The candidate set of predictive class pairs $\mathbf{C_{cand}}$, the training dataset $D$.
    $\mathbf{C}^{\dagger} \leftarrow \emptyset$
    Randomly partition $D$ into $D_{learn}$ and $D_{valid}$.
    for Each $(C_i \leftharpoonup C_{i'}) \in \mathbf{C_{cand}}$ do
        $predsPair \leftarrow \emptyset$
        $predsSingle \leftarrow \emptyset$
        Generate $D^{\tilde{c_i}}_{(i,i')}$ and $D^{c_i}_{(i,i')}$ from $D_{learn}$ using $C_{i'}$ to split the data and the $F$-test
        algorithm to select the most relevant features.
        Induce $M^{\tilde{c_i}}_{(i,i')}$ using the features selected from $D^{\tilde{c_i}}_{(i,i')}$.
        Induce $M^{c_i}_{(i,i')}$ using the features selected from $D^{c_i}_{(i,i')}$.
        Induce $M_{(i,i')}$ using the features selected from $D_{learn}$.
        for Each instance $(\mathbf{x}', C') \in D_{valid}$ do
            if $C'_{i'} = c_i$ then
                $predsPair \leftarrow predsPair \cup M^{c_i}_{(i,i')}(\mathbf{x}')$
            else
                $predsPair \leftarrow predsPair \cup M^{\tilde{c_i}}_{(i,i')}(\mathbf{x}')$
            end if
            $predsSingle \leftarrow predsSingle \cup M_{(i,i')}(\mathbf{x}')$
        end for
        if $AUPRC(predsPair) > AUPRC(predsSingle)$ then
            $\mathbf{C}^{\dagger} \leftarrow \mathbf{C}^{\dagger} \cup (C_i \leftharpoonup C_{i'})$
        end if
    end for
    Induce graph $G$, treating the pairs $(C_i \leftharpoonup C_{i'}) \in \mathbf{C}^{\dagger}$, as directed edges from
    $C_{i'}$ to $C_i$.
    return $G$ and $\mathbf{C}^{\dagger}$.
end procedure
```

we use the classifier that was trained to predict C_i only using instances in D with $C_{i'} = \tilde{c_{i'}}$. This procedure is repeated for every classifier pair, one pair for each class variable in $\mathbf{C_i^{CB}}$, giving us several estimations for $P(C_i|\mathbf{C_i^{FB}}, C_{i'})$, one for each class variable $C_{i'}$ in $\mathbf{C_i^{CB}}$. To return the unified estimation of $P(C_i|\mathbf{C_i^{FB}}, \mathbf{C_i^{CB}})$ that the Gibbs sampling algorithm requires, we average the estimations of $P(C_i|\mathbf{C_i^{FB}}, C_{i'})$ given by each one of the classifiers corresponding to the variables on the Markov blanket of C_i, that is, the average over the class probabilities in the set *predictions* in Algorithm 5.3.

Algorithm 5.3 Estimating $P(C_i|\mathbf{x}, \mathbf{C_i^{CB}})$.

```
procedure ESTIMATION OF $P(C_i|\mathbf{x}, \mathbf{C_i^{CB}})$
Inputs: the instance's feature vector $\mathbf{x}$, the class variable to be predicted $C_i$, the Dependence Network $G$, the current predictions of classes other than $C_i$ for the instance $\mathbf{C}^{\dagger}$.
    $\mathbf{C_i^{CB}} \leftarrow$ The class variables that point to $C_i$ in $G$.
    predictions $\leftarrow \emptyset$
    for Each class variable $C_{i'} \in \mathbf{C_i^{CB}}$ do
        if $C_{i'} = c_{i'}$ then
            $\mathbf{C_{i'}^{FB}} \leftarrow$ the features selected by the $F$-test from $\mathbf{x}$ when $C_{i'} = c_{i'}$.
            predictions $\leftarrow$ predictions $\cup M^{c_{i'}}_{(i,i')}(\mathbf{C_{i'}^{FB}})$
        else
            $\mathbf{C_{i'}^{FB}} \leftarrow$ the features selected by $F$-test from $\mathbf{x}$ when $C_{i'} = \tilde{c_{i'}}$.
            predictions $\leftarrow$ predictions $\cup M^{\tilde{c_{i'}}}_{(i,i')}(\mathbf{C_{i'}^{FB}})$
        end if
    end for
    return the average class probability of the individual probabilities in predictions
end procedure
```

The HDN-nHPC variation of the HDN algorithm uses the predictive class variables in very a different way than the previous versions. The HDN and HDN-PCT algorithms treat the class variable values in the Markov blanket of the class nodes as common features (see Algorithm 5.1), while the HDN-nHPC treats the values of the class variables as special features that guide the decision of which classifier in the classifier pairs to use. The simpler approach of using the class variables as features could, in principle, be used in the HDN-nHPC algorithm as well, but note that that Algorithm 5.2, used to find the set of predictive class relationships, works by finding classifier pairs with better predictive accuracy than a single classifier. Therefore, in the HDN-nHPC algorithm, it arguably makes more sense to use the classifiers in the pair, which are known to be good predictors of C_i, than using the predictive class variables as common features. The superiority of using classifier pairs in relation to using predictive classes as features (like in the HDN algorithm) in our HDN-nHPC algorithm was confirmed in preliminary experiments.

To query our Dependence Network we use a second version of the modified Gibbs sampling, presented in Algorithm 5.4. The main difference between this algorithm and Algorithm 5.1 is on how to use the predictions in the Markov blanket of the class variables - the class blanket predictions. While Algorithm 5.1 appends these predictions to the feature vector of the instances, using these class predictions as new features (lines 5 to 7), Algorithm 5.4 passes the value of the predictions as a parameter to Algorithm 5.3 (line 10), which in turn calculates the probability $P(C_i|\mathbf{x}, \mathbf{C_i^{CB}})$ using the classifier pairs, as previously described.

If the set $\mathbf{C_i^{CB}}$ is empty for some class variable (either because there is no edge in G from some other variable to C_i, or because C_i is not in G), we use a standard SVM classifier model (denoted as $M_i(\mathbf{C_i^{FB}})$) to estimate $P(C_i|\mathbf{C^{FB}}_i)$. In this case, there is no need to run the Gibbs sampling procedure for this particular class variable, as its class probability distribution does not depend on the presence of any other class variable.

After all posterior probabilities are calculated, we traverse the class hierarchy in a bottom up-fashion, checking if each class probability is greater than or equal to the class probability of its descendants. If this is not the case, we set the class probability of the offending class to be the maximum probability of its descendants (Obozinski, Lanckriet and Grant 2008).

We call the conjunction of Algorithms 5.2, 5.3 and the modified Gibbs sampling algorithm (Algorithm 5.4): HDN based on finding non-Hierarchically related Predictive Classes (HDN-nHPC).

Now that we have precisely defined the HDN-nHPC algorithm, we can clarify the characteristics of this algorithm that distinguish it from the one proposed in Section 5.2 (HDN):

1. The data-driven approach to find predictive class relationships is performed in Algorithm 5.2 by building the graph G. This graph represents important predictive class relationships among classes that are non-hierarchically related (i.e., classes that are not ancestors or descendants of each other). By contrast, the HDN algorithm simply uses the siblings and parents of the children of the current class as predictive features, ignoring the vast majority of the class hierarchy when predicting the current class.

Algorithm 5.4 Hierarchical classification with Modified Gibbs sampling algorithm.

```
procedure Modified Gibbs sampling
Inputs: Instance feature vector $\mathbf{x}$, total number of iterations $it$, number of burn-in iterations $burn$.
    Probabilistically initialise the predicted class label vector of $\mathbf{x}$ ($\mathbf{C_{pred}}$), using uniform distributions (parametrised by the a priori empirical class distributions).
    for $k \in \{1..it\}$ do
        for all $C_i \in$ the Class Taxonomy do
            $\mathbf{C_i^{CB}} \leftarrow$ Get classifications of the Markov blanket of $C_i$ from $\mathbf{C_{pred}}$.
            if $\mathbf{C_i^{CB}} = \emptyset$ (No class affects the prediction of $C_i$) then
                $P(C_i|\mathbf{x}) \leftarrow M_i(\mathbf{C_i^{FB}})$ (Use a flat classifier to estimate $P(C_i|\mathbf{x})$.)
                Skip the rest of the “for all” loop for the current $C_i$.
            end if
            $p \leftarrow P(C_i|\mathbf{x}, \mathbf{C_i^{CB}})$ /* Call Algorithm 5.3*/
            $u \leftarrow$ Random value drawn from a uniform distribution in $[0,1]$.
            if $u < p$ then $C_{pred,i} \leftarrow c_i$ else $C_{pred,i} \leftarrow \tilde{c}_i$ end if
            if $k > burn$ then
                $P(C_i|\mathbf{x}) \leftarrow \frac{P(C_i|\mathbf{x}) \times (k - burn - 1) + p}{k - burn}$
            end if
        end for
    end for
    return the marginal probabilities $P(C_i|\mathbf{x})$.
end procedure
```

2. The dataset split approach to induce classifiers is performed in Algo. 5.2 by the creation of different classification models ($M^{\tilde{c_i}}_{(i,i')}$ and $M^{c_i}_{(i,i')}$) using different subsets of the training set ($D^{\tilde{c_i}}_{(i,i')}$ and $D^{c_i}_{(i,i')}$), by splitting the training set based on the two values of a predictive class variable. By contrast, the HDN algorithm uses the values of the class variables in the Markov blanket of each node as a common predictive attribute to induce the base classifiers.

3. The use of several classification models to estimate the probability of a class is done in Algorithm 5.3 by averaging the predictions of the classifiers in the Markov blanket of each class label. By contrast, the HDN algorithm estimates the probability of a class by using a single classification model using the class variables in the Markov blanket as extended features.

5.5 The Hybrid Predictive Clustering Tree/Local Hierarchical Classification (PCT-LHC) Algorithm

This hybrid hierarchical classification algorithm can be seen as an extension of the PCT algorithm, which builds a global model where each leaf node is assigned a class probability vector. The PCT model simply assigns, to a new instance reaching a given leaf node, the classes whose probabilities are greater than a certain threshold. Our new hybrid algorithm first uses the standard PCT algorithm to build the model (decision tree) in the training phase. Next, for each leaf node having more than *min_inst_LHC* (a parameter) instances, the hybrid algorithm builds a local classification model for predicting each class, by running a standard *flat* classification algorithm from the instances in that leaf node. The idea is that leaves with a relatively large number of instances may be further explored by another classification algorithm, in this case the LHC algorithm, hopefully improving predictive performance. Again, we used a SVM as the local (base) classification algorithm in the training phase.

In the testing phase, when an instance is presented to the PCT classification model the leaf node associated with that instance is identified. If flat classifiers were trained in the leaf node, they are used to predict the class labels of the instance. If the flat classifiers were not trained in the leaf node, the class probability vector of the PCT is returned, as usual.

The combination of decision trees with other classification algorithms has been recently proposed for (flat) multi-label classification with success (Gjorgjevkikj, Madjarov and Dzeroski 2013); however, as far as we know, it was first proposed for hierarchical classification by us in (Fabris, Freitas and Tullet 2015).

5.6 Conclusions

In this chapter we have proposed four new hierarchical classification algorithms. The first one, the HDN (defined in Section 5.2), uses the structure of the class hierarchy to define the edges of a DN (the Markov blanket of each node) and a single local 'flat' classifier to estimate the probability of each class in the hierarchy, using the state of the Markov blanket of that class as extended predictive features. This algorithm uses the Gibbs sampling approach to get the final prediction of the HDN model.

The second hierarchical classification algorithm we have proposed is the hybrid HDN-PCT algorithm (defined in Section 5.3), which induces a HDN classifier in each cluster formed by the PCT algorithm.

The third hierarchical classification algorithm we have proposed is the HDN-nHPC algorithm (defined in Section 5.4). This algorithm has a different approach to build the DN. Instead of using the given hierarchical structure, this algorithm uses a data-driven approach to find relationships among classes that are not encoded directly by the class hierarchy and (potentially) induces more than one 'flat' classifier to predict each class in the hierarchy. This classifier uses a modified Gibbs sampling algorithm to combine the predictions of each 'flat' classification model induced to predict each class label.

The fourth algorithm we have proposed is a hybrid between the PCT classification algorithm and the LHC classification algorithm (defined in Section 5.5), two popular algorithms for hierarchical classification. This hybrid is similar to the HDN-PCT, the difference is that a LHC classifier is induced in each cluster formed by the PCT algorithm, instead of a HDN classifier.

Chapter 6

Results for the Hierarchical Classification Algorithms

6.1 Hierarchical Classification Datasets Used in this Work

In this section we present the three groups of datasets we used for our hierarchical classification experiments. Subsection 6.1.1 presents details of the hierarchical datasets made publicly available in `https://dtai.cs.kuleuven.be/clus/hmcdatasets/` by Celine Vens (Vens et al. 2008), which use *FunCat* (Ruepp et al. 2004) and *Gene Ontology* (Harris et al. 2004) hierarchical terms as predictive classes. Subsection 6.1.2 describes how we created ageing datasets for several model organisms using terms of the *Gene Ontology* hierarchy as predictive classes. Subsection 6.1.3 describes how we created ageing datasets for the mouse model organism using *Mammalian Phenotype Ontology* (Eppig et al. 2015) hierarchical terms as predictive classes. In Subsection 6.1.4 we present general characteristics of all datasets we have considered.

6.1.1 Vens' Datasets

The first set of datasets we used are 22 of the 24 datasets made available in `https://dtai.cs.kuleuven.be/clus/hmcdatasets/`. We discarded the datasets pheno_GO and pheno_FUN, since they contain many (more than 50%) missing values, and adding them would require a non-trivial adaptation of the hierarchical classifiers we used. The datasets are pre-divided by their creators into training, validation and testing sets; we maintain the same division in our work. We did not employ the more traditional cross-validation procedure because of the relatively large number of instances and because the use of the same dataset division as in (Vens et al. 2008) makes our results more fairly comparable with other works in the literature.

These datasets contain features extracted from the genes of the widely used model organism *Saccharomyces cerevisiae* (yeast). There are two types of predictive features: 1) statistics extracted from the amino acid sequences (seq features) and 2) several types of microarray expression data (all the other features). (Clare and King 2003) explains the creation of the features in detail.

There are also two types of class hierarchies: "FUN" (the tree-structured hierarchy in the FunCat scheme (Ruepp et al. 2004)) and "GO" (the DAG-structured Gene Ontology (Harris et al. 2004)).

The characteristics of these datasets will be presented later on, in Table 6.3.

6.1.2 Ageing-Related Gene Ontology (GO) Datasets

To study the biological aspects of ageing/longevity using our hierarchical classification algorithms, we have built 15 datasets containing features extracted from the proteins encoded by the genes in the Ageing Gene Database (GenAge) (Tacutu et al. 2013). GenAge is a catalog of ageing-related genes coming from several species, including human and model organisms such as *S. cerevisiae* (baker's yeast) and *M. musculus* (the house mouse).

Salama and Freitas (2013) have already compiled an ageing-related dataset for the hierarchical classification of ageing-related proteins. We build upon their work by updating and expanding the dataset to contain more species and the features used in (Silla Jr. and Freitas 2011b), which focused on the hierarchical classification of generic (not specifically ageing-related) proteins functions. In our datasets, each instance represents an ageing-related gene, and the hierarchical classes to be predicted are Gene Ontology (GO) terms.

All genes were collected from the GenAge database, build 17 (from December 18, 2013). This version contains 298 human ageing-related genes and 1,825 genes from model organisms, related to both ageing and longevity.

The human gene dataset contains a comprehensive list of genes potentially associated with human ageing. This list contains genes supported by different degrees of confidence, varying from direct evidence linking the gene to human ageing

Table 6.1: Number of genes and proteins for each species present in the GenAge database.

Species	Number of genes	Number of proteins
Saccharomyces cerevisiae (baker's yeast)	825	762
Caenorhabditis elegans (a type of worm)	741	263
Homo sapiens (human)	298	301
Drosophila melanogaster (fruit fly)	140	79
Mus musculus (house mouse)	112	107
Podospora anserina (a type of filamentous fungus)	3	-
Schizosaccharomyces pombe (Fission yeast)	2	-
Danio rerio (Zebra fish)	1	-
Mesocricetus auratus (Golden hamster)	1	-

to inconclusive evidence that the gene is related to ageing. The model organism datasets contain genes associated with ageing in non-human organisms. These genes have, in general, a more reliable classification due to easiness of experimenting with these species. Table 6.1 lists in the second column the number of genes for each organism in GenAge.

Because of the low representativity, we discard the species *P. anserina*, *S. pombe*, *D. rerio* and *M. auratus*. This leaves us with five species. For each species we derive three datasets containing three broad types of features, namely numeric alignment independent features, protein motif features and protein-protein interaction features, leaving us with 15 ageing-related datasets.

The GenAge database contains the *"Entrez Gene Id"* as an external gene identifier; we use it to retrieve the *"UniprotKB AC ID"* (UniProt Knowledge Database Accession Identifier) protein identifier using the UniProt ID Mapping Tool.

Because more than one protein may be associated with a single gene, 2,855 UniProt identifiers were retrieved from the 2,123 genes. However, from the 2,855 proteins, we discard 1,243 whose functions were not manually reviewed by experts or whose species is one of the four that were discarded. After this step, we end up with 1,612 proteins (instances), distributed among organisms as presented in the last column of Table 6.1.

Finally, we downloaded the amino acid sequence of each protein from the UniProt-SwissProt database, using build 2014_02 of 19 February 2014[1]. We describe below how we created the features for our datasets.

[1] ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/

The hierarchical classes were created for each model organism by first retrieving the GO terms associated with each protein sequence using the UniProt-SwissProt database. Next, we used the DAVID tool[2] (version 6.7 of January of 2010) to retrieve the over-expressed GO terms of each model organism, considering only these GO terms in our final dataset. We call these over-expressed GO terms ageing-related GO terms, as they occur significantly more often than statistically expected in our datasets of ageing-related proteins.

Note that there is currently a new version of the DAVID tool (version 6.8 of October of 2016) which updated the data used by DAVID in several ways, including the definition of the ontology used to calculated the over-expressed GO terms. The task of updating the data mining dataset used in this work is left as future work. Researchers interested in using the datasets compiled for this work should be careful, since this data is now somewhat obsolete.

The characteristics of these datasets will be presented later on, in the first part of Table 6.4.

Numeric Alignment-Independent Features

We extracted the following numeric features described in (Salama and Freitas 2013; Silla Jr. and Freitas 2011a): “Amino Acid Composition” (21 features), “Composition” (3 features), “Transition” (3 features), “Distribution” (15 features), and “Z-Values” (15 features). Furthermore, all datasets (with all types of features) have two features: “Sequence Length” (the amino acid sequence length), and “Molecular Weight” (the molecular weight of the protein). These features are called alignment-independent, as they do not require any alignment procedure such as “BLAST” or“PSI-BLAST” to be performed on the sequences prior to their calculation.

The ‘BLAST‘ algorithm works by finding similar sequences to the query sequence by counting the number of ‘operations’ (deletions, substitutions and insertions) necessary to transform one sequence into the other. The BLAST algorithm uses a pre-computed cost matrix to measure the ‘cost’ of each operation, for each amino acid pair. The ‘PSI-BLAST’ algorithm, on the other hand, runs several ‘BLAST’ queries, updating the cost matrix according to the current set of retrieved ‘similar’ sequences, thus changing the cost matrix dynamically (Bhagwat and Aravind 2007). The idea behind this is to update the substitution costs empirically,

[2] http://david.abcc.ncifcrf.gov

taking into account conserved and non-conserved changes in the protein sequences (changes in conserved sites are probably more relevant, as they are more likely to change protein function). Note that this dynamically computed cost matrix can be analysed to gain insights of important regions of the protein.

It is desirable to avoid such alignment algorithms as they usually require parameter setting, usually rely on the assumption that similar sequences share similar functions (which is not always the case), and may be very computationally intensive.

Each "Amino Acid Composition" feature is the ratio of the frequency of a distinct amino acid in the sequence over the total number of amino acids in the sequence. Each "Composition" feature is the frequency of each type of amino acid (polar, apolar or neutral) in the amino acid sequence. "Transition" features measure how common are the transitions among amino acid types (polar $\leftrightarrow$ apolar, polar $\leftrightarrow$ neutral, and apolar $\leftrightarrow$ neutral) by calculating their relative frequency. "Distribution" features measure the relative location in the amino acid sequence of the first, the first 25%, the first 50%, the first 75% and all amino acids of a given functional group (polar, apolar or neutral). "Z-Values" are features derived from the 5 principal components (produced by principal component analysis) of several physiochemical properties of each amino acid (Sandberg et al. 1998). Following the suggestion of (Secker et al. 2007), the five Z-values of each amino acid in the sequence are averaged across the whole sequence, across the first 100 amino acids, and across the last 100 amino acids, creating a total of 15 "Z-value" features. We have used the z-values presented in Table 6.2, extracted from Sandberg et al. (1998).

In (Secker et al. 2007) several ways to combine the z-values of the amino acid sequence were tested. Authors concluded that the z-values should be averaged across the whole amino acid sequence. In their work, which had the goal of classifying GPCR (G-Protein Coupled Receptor) proteins, the C-terminus and N-terminus of the proteins (the extremities of the molecule) are of great importance; therefore, besides using five z-values averaged over the whole sequence, the authors recommend using two sets of five z-values averaged over the first 150 and last 150 amino acids of the protein sequence as well. Although we are not classifying the same type of protein, we decided to test the effectiveness of these extra 10 z-values as attributes too.

In addition, following (Freitas, Vasieva and de Magalhães 2011), we extend each of the human ageing datasets with the D_n/D_s ratio, which measures the

Table 6.2: Z-values extracted from Sandberg et al. (1998)

Amino Acid	z_1	z_2	z_3	z_4	z_5
A	0.24	-2.32	0.60	-0.14	1.30
C	0.84	-1.67	3.71	0.18	-2.65
D	3.98	0.93	1.93	-2.46	0.75
E	3.11	0.26	-0.11	-3.04	-0.25
F	-4.22	1.94	1.06	0.54	-0.62
G	2.05	-4.06	0.36	-0.82	-0.38
H	2.47	1.95	0.26	3.90	0.09
I	-3.89	-1.73	-1.71	-0.84	0.26
K	2.29	0.89	-2.49	1.49	0.31
L	-4.28	-1.30	-1.49	-0.72	0.84
M	-2.85	-0.22	0.47	1.94	-0.98
N	3.05	1.62	1.04	-1.15	1.61
P	-1.66	0.27	1.84	0.70	2.00
Q	1.75	0.50	-1.44	-1.34	0.66
R	3.52	2.50	-3.50	1.99	-0.17
S	2.39	-1.07	1.15	-1.39	0.67
T	0.75	-2.18	-1.12	-1.46	-0.40
V	-2.59	-2.64	-1.54	-0.85	-0.02
Y	-2.54	2.44	0.43	0.04	-1.47
W	-4.36	3.94	0.59	3.44	-1.59

degree of conservation between two gene sequences (Yang and Bielawski 2000). D_n is the ratio of non-silent mutations between two sequences (gene mutations that alter the amino acid sequence of the protein associated with that gene), D_s is the ratio of silent mutations (changes in the gene sequence that do not alter the corresponding protein). We compare the D_n/D_s ratio of homolog ageing-related genes in humans and rhesus macaque (*Macaca mulatta*). Larger ratios indicate a recent evolutionary change in that gene, suggesting a stronger relation with ageing than genes with smaller values. Using the D_n/D_s ratios from the BioMart tool[3] we extracted 288 D_n/D_s ratios from the human/rhesus genes. 8 genes had no homologs in the *Homologene* dataset and have missing values for this D_n/D_s feature in the datasets.

Protein-Protein Interaction (PPI) Features

This type of binary feature indicates whether or not an ageing-related protein interacts with each of a set of other proteins (which may or may not be ageing-related proteins). Interacting partners of one protein often give away hints of its function (Sharan, Ulitsky and Shamir 2007). This type of feature was recently used in ageing-related datasets (Freitas, Vasieva and de Magalhães 2011). We have used the BioGrid[4] database to extract PPIs and have only considered features representing interacting partners occurring in three or more instances in the dataset, to avoid classifier over-fitting due to the rare interactions with a given protein.

Protein Motif Features

The binary motif features represent the presence or absence of a given motif in the amino acid sequence of a protein. A motif is a template describing similar sequences of amino acids that occur recurrently in proteins. Motifs serve as a high-level representation of a protein and it is expected that proteins sharing some specific motifs share similar functions. We have used the same four types of motifs investigated in (Silla Jr. and Freitas 2011a): Interpro (Hunter et al. 2012), Pfam (Finn et al. 2008), Prosite (Sigrist et al. 2013), PRINTS (Attwood et al. 2003). We have only considered motifs occurring in at least three proteins (instances) in the dataset, to avoid overfiting as mentioned earlier.

[3] `http://www.ensembl.org/biomart/`

[4] http://thebiogrid.org

6.1.3 Ageing Mammalian Phenotype Ontology (MPO) Datasets

To explore different predictive features and predictive classes, we have built 5 datasets containing features extracted from the proteins encoded by the genes in the *Phenotypes and Mutant Alleles* section of the *Mouse Genome Informatics* (MGI) database. The MGI provides the two primary sources of data for our datasets: (1) the definition of the *Mammalian Phenotype Ontology* (MPO), the source of class labels to be predicted, and (2) a list of genotypes annotated with the phenotypes present in the MPO, the source of the features (predictors).

The MPO is organised as a DAG (Directed Acyclic Graph), where each node represents a phenotype (an ontology term) and each edge an “IS-A” relation between phenotypes.

The MPO contains 10,907 terms in total, and 113 terms under the term MP:0010768 (ageing/mortality) part of the hierarchy, our research focus. We consider only the 113 ageing-related terms as class labels for our study, and discard the others. Considering all 10,907 terms would generate classification models more focused on predicting non-ageing-related terms, generating models with less interest for the biology of ageing. After further discarding MPO terms with less than 10 instances, we end up with 81 MPO terms, the hierarchical class labels to be predicted.

With the class hierarchy defined, we must create our instances. In the MGI database, 11,532 genotypes are annotated with at least one of the 113 mortality/ageing-related ontology terms. Each genotype is formed by a list of allele mutations. Each allele mutation contains (among other information) one or more protein-encoding genes, which in turn are associated with particular mutations. Therefore, using the MPO hierarchy we can associate a protein (instance) with one or more phenotypes (hierarchical classes). Figure 6.1 shows these relations graphically.

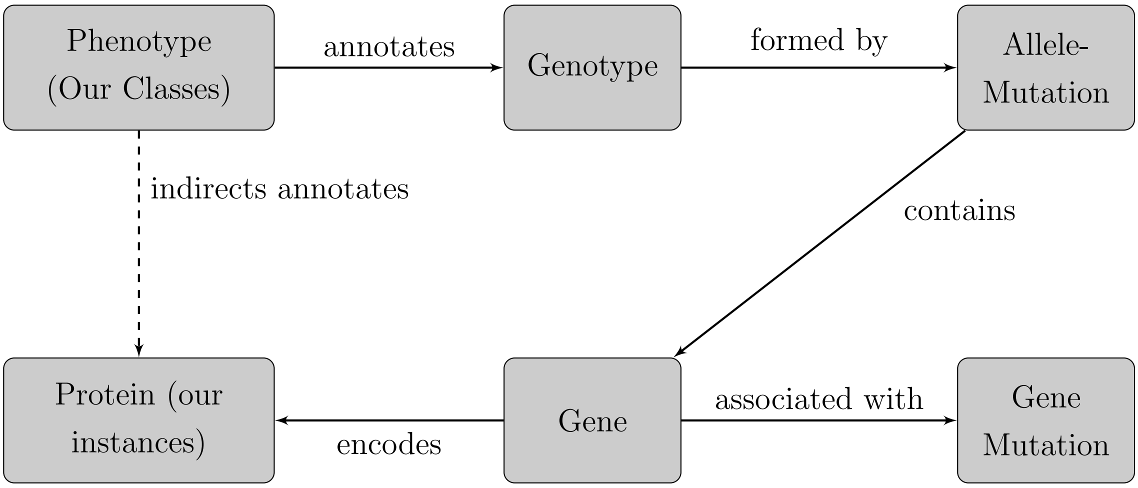


Figure 6.1: Relationships among MPO elements and the instances in our datasets. Filled edges represent relationships present in the MGI database. The dashed edge represents the indirect relation that we use for our datasets. Note that we ignore mutation information.

Note that our instances are proteins encoded by standard genes, not gene mutations, because, as discussed later, information about proteins is much richer and precise than information about gene mutations.

However, choosing to use proteins as instances (instead of gene mutations) has the disadvantage of risking annotating the same protein with contradictory MPO terms. This may happen because two different mutations on the same gene may have contradictory effects. E.g., one mutation may over-express the protein encoded by a gene, while some other mutation on the same gene may under-express that protein, possibly leading to opposite MPO terms being associated with the same gene with different mutations. Since this mutation information is not available for the classifier, these apparent contradictions (opposite MPO terms annotating the same instance) may reduce classification performance and interpretability. However, we consider this compromise acceptable since the lack of information about particular gene mutations makes the use of classification algorithms considering gene mutations as instances inviable.

Following this approach, we merged the annotations associated with the same gene, keeping all MPO terms that were associated with the different mutations of that gene. After this step, the 11,532 gene-mutations were reduced to 5,045 genes (without mutation information) keeping all annotations associated with different gene mutations.

The next step is to retrieve the Entrez Id (unique gene identifier) for each one of the 5,045 genes associated with the mortality/ageing phenotypes. Genes without an Entrez Id were discarded, further reducing the number of instances to 4,575. Finally, we retrieved the UniProt Id associated with each Entrez Id., using the UniProt ID Mapping Tool. This gives us information about the protein product associated with each gene. Genes having the same UniProt Id were discarded, leaving us with the final number of 3,886 proteins (instances), each instance linked with one protein and a list of mortality/ageing phenotypes (MPO terms used as class labels).

For the list of 3,886 proteins, we derive five datasets, each with a different feature type: numeric features, protein motifs features, Protein-Protein Interaction (PPI) features (explained in Subsection 6.1.2), and two types of KEGG pathway features, described next.

KEGG Pertinence (KEGGP) pathway features - KEGG pathways are directed-graph representations of interactions between several types of biological products (e.g., genes or proteins) (Kanehisa et al. 2016). To build our KEGG pathway features we have parsed the KGML representations of the mouse KEGG pathways under the condition that at least 1% of our instances must be present in the pathway in order for the pathway to be considered. This generated a total of 221 KEGGP features. Pertinence features based on KEGG pathways have already been explored in other works involving 'flat' data mining, e.g., (Jungjit et al. 2014; Keerthikumar et al. 2009), but not in hierarchical classification yet.

KEGG Influence (KEGGI) pathway features - This feature type, proposed in (Fabris and Freitas 2016), quantifies the influence that an instance (reference protein) has on the downstream proteins of a KEGG pathway, the idea being that proteins that have a common influence on a set of downstream protein share similar function. Consider that one ageing-related protein affects a set of downstream proteins in a given way. If another protein affects the downstream proteins in a similar way, then it is likely that that protein is also ageing-related.

The use of complex KEGG-based pathway features for data mining has been proposed in other works, as follows: (Zhang and Wiemann 2009) proposed a software tool to construct a graph-based model of KEGG pathways. (Xia and Wishart 2010) used graph-based KEGG features for *metabolomics analysis*. (Chen et al. 2010) used characteristics extracted from the KEGG pathway graph to classify the pathways into "biologically meaningful" or not. (Breitkreutz et al. 2012) correlated the complexity of cancer-related KEGG pathways to patient survivability. Despite being previously used for different goals, as far as we know, we are the first to propose complex KEGG-based features for the classification of protein functions.

The influence score for a given protein p has the minimum value of 0.0 when the reference protein (P_{ref}) does not influence p at all, because p is not "downstream" of (i.e., cannot be reached from) P_{ref}.

Figure 6.2 shows an example of the calculation of the proposed "influence" score for a hypothetical instance (reference protein) and a set of downstream proteins. Proteins P_1, P_2 and P_6 in that figure have a score of 0.0, since they are not downstream of P_{ref}.

The score of a given protein p that is downstream of P_{ref} has the maximum value of 1.0 if, when P_{ref} is removed from the pathway, the downstream protein p becomes unreachable from the proteins that are not downstream proteins of P_{ref}. The biological meaning that we want to capture is that a knockout on P_{ref} would nullify the standard behaviour of the downstream protein p. Proteins P_3 and P_7, in Figure 6.2, have a score of 1.0 since if P_{ref} is removed from the pathway, proteins P_3 and P_7 will be disconnected from the KEGG pathway graph defined by the set of proteins that are not downstream proteins.

If the score of a given protein p that is downstream of P_{ref} has a value of 0.5, it means that P_{ref} accounts for half of the influence that the downstream protein p receives. Removal of P_{ref} would not completely nullify the standard behaviour of the downstream protein p, because there would be one more protein (which is not downstream in relation to P_{ref}) that also affects p, therefore the influence of P_{ref} on p is 50%. Protein P_5, in Figure 6.2, has a score of 0.5 because if one removes protein P_{ref} from the graph, protein P_5 would still be reachable from protein P_2, which is not a downstream protein.

In practice, to calculate the value of the features for each instance, we need to build two sets of proteins: the first, the *downstream proteins*, comprises the proteins that are downstream of the current instance, P_{ref}. The second set, the *non-downstream parent proteins*, contains the proteins that are not downstream of P_{ref} but are the parents of a protein that is downstream of P_{ref} - e.g., proteins P_2 and P_6 in Figure 6.2. Finally, for each downstream protein, the influence score is equal to $1/(1 + p_{effect})$, where p_{effect} is the number of non-downstream parent proteins that have an effect (direct or indirect) on the downstream protein. We

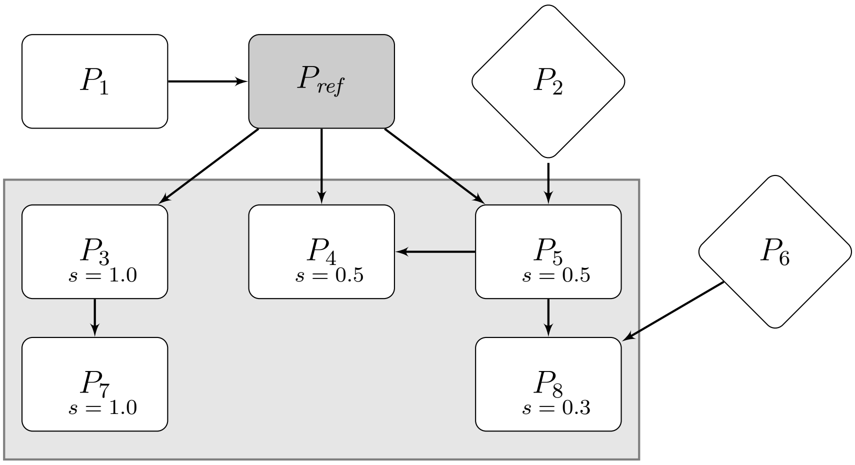


Figure 6.2: Example of score values (s) for five downstream proteins (P_3, P_4, P_5, P_7, P_8) in relation to a reference protein P_{ref}. Diamond-shaped nodes represent proteins that are parent of some downstream protein but are not downstream protein themselves.

consider that a non-downstream protein has an effect on the downstream protein if the non-downstream proteins can reach the downstream protein.

To illustrate these concepts in detail lets us consider protein P_8 (see Figure 6.2), which is in the set of downstream proteins of P_{ref}. Because both non-downstream parent proteins affect P_8 (both P_2 and P_6 can reach P_8), the value of the influence score for P_8 is $1/(1 + 2) = 0.3$.

This gives us a set of downstream protein scores for the instance. We repeat this procedure for every available KEGG pathway. If the same downstream protein occurs more once in the same pathway, we keep the highest score. We discard the features (downstream proteins) with value > 0.0 in less than 1% of the instances, totalling 618 features. We call the KEGG pathway influence features KEGGI from now on.

6.1.4 Hierarchical Dataset Statistics

Tables 6.3 and 6.4 contain information about the datasets we are using to test our algorithms. In particular, we can see in Table 6.3 that the Vens' datasets have a relatively high mean number of instances (3672.5) and GO predictive classes (3649.5).

The mean number of FunCat classes is much smaller (473.3), this is natural, as the FunCat hierarchy has much fewer terms than the GO; therefore, it is expected that a smaller number of terms annotate each instance.

Table 6.3: Number of instances, predictive features and classes in the Vens' datasets used in this work. Note that although the table has 11 entries, the total number of hierarchical datasets is 22, 11 for each type of class hierarchy (GO and FunCat).

Predictive Features	Number of FunCat Classes	Number of GO classes	Number of Instances	Number of Features
seq	476	3704	3932	478
cellcycle	476	3695	3766	77
church	476	3696	3764	27
derisi	476	3691	3733	63
eisen	447	3176	2425	79
gasch1	476	3698	3773	173
gasch2	476	3698	3788	52
spo	476	3691	3711	80
expr	476	3698	3788	551
struc	476	3703	3851	19628
hom	476	3695	3867	47034

The number of features is the dataset characteristic with the highest variance in Table 6.3. This is due to the different representations that were used to create the datasets, specially those coming from microarray expression data, which can vary greatly in size.

Table 6.4 presents the characteristics for the datasets we created. We can see that the number of instances for these datasets varies considerably more, due to the fact that they come from different animal species and dataset sources.

The number of distinct hierarchical classes in these datasets also varies more than for the datasets in Table 6.3. This is due, first, to the varying number of instances, and second to the different animal species, which are annotated with varying degrees of detail.

The number of features also varies significantly in Table 6.4, since datasets with more instances tend to naturally have more features (there is more chance for more non-zero attributes being present) and different species have varying numbers of Motif and PPI annotations.

Table 6.4: Number of hierarchical class labels, instances, and predictive features in the datasets we created and used in this work.

Hier.	Feature Type		Number of Hier. Classes	Number of Instances	Number of Features
Ageing GO	worm	Numeric	350	263	59
		PPI			162
		Motifs			112
	fly	Numeric	385	79	59
		PPI			105
		Motifs			55
	human	Numeric	1713	301	60
		PPI			2425
		Motifs			284
	mouse	Numeric	683	107	59
		PPI			29
		Motifs			40
	yeast	Numeric	583	762	59
		PPI			4397
		Motifs			296
Ageing MPO	KEGG		84	3886	221
	KEGGI				618
	Motifs				372
	Numeric				59
	PPI				123

6.2 Predictive Performance Measures

As the algorithms we are testing output the probabilities of an instance belonging to each class, instead of a crisp classification, to estimate their predictive performance, we could transform the class probabilities into crisp classifications by predicting classes whose probability is greater than a certain threshold - that is, a given class is assigned to the current test instance if and only if the output probability for that class is greater than the used threshold.

In order to avoid the subjective choice of a threshold, in binary classification, it is common to calculate the Area Under the Precision Recall Curve (AUPRC). In the case of highly skewed class distributions (as is typically the case of hierarchical classification problems), the AUPRC is more appropriate than the more popular Area Under the Receiver Operating Characteristic Curve (AUROC) because the

AUROC measure over-rewards classifying negative instances as negative (Davis and Goadrich 2006), not recognising that correctly classifying a rare positive instance in the highly negatively-skewed scenario should be more important than correctly classifying the more common negative instances.

Precision (P) and Recall (R) are common measures used in binary classification problems, defined as:

$$P \equiv \frac{TP}{TP+FP}, \quad \text{and} \quad R \equiv \frac{TP}{TP+FN}. \tag{6.1}$$

Where TP (True Positives) is the number of correctly positively classified instances, FP (False Positives) is the number of instances classified as positive that are not positive, FN (False Negatives) is the number of positive instances that were not classified as positive and "≡" means "equal by definition".

These two measures represent conflicting optimising goals. E.g., one can obtain a high precision (but a low recall) by classifying as positive only instances that have a high probability of belonging to the positive class; conversely, one can obtain a high recall (and a low precision) by classifying all instances as positive.

Given a binary classifier with probabilistic outputs, it is possible to construct a PR curve (a plot of the classifier's precision as a function of its recall) by thresholding the output (class probability) of the classifier using values in the interval $[0, 1]$. Each threshold is associated with a value of precision and recall, corresponding to a point in the PR space. Note that to obtain a single performance measure from the curve, we calculate its area using a trapezoidal approximation (Boyd, Eng and Page 2013). A classifier that perfectly ranks all negative instances before the positive ones – where the ranking is by increasing order of probability of being positive – would have a $AUPRC$ of 1.0.

There is no obvious way to adapt this measure to the hierarchical classification scenario, therefore we follow the suggestions of (Vens et al. 2008) and use three alternatives: the Area Under the average Precision-Recall Curve ($AU(\overline{PRC})$), the average Area Under the Precision Recall Curve ($\overline{AUPRC}$) and the weighted average Area Under the Precision Recall Curve ($\overline{AUPRC_w}$).

To calculate the $AU(\overline{PRC})$, we use the hierarchical versions of precision and recall for a fixed threshold, defined as:

$$hP \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |P_j|} \quad \text{and} \tag{6.2}$$

$$hR \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |T_j|}. \tag{6.3}$$

Where P_j is the set of predicted classes of the j-th instance and T_j is the set of true classes of the j-th instance.

To calculate the $\overline{AUPRC}$ measure we simply average all the class-wise $AUPRC$ values. Similarly, to calculate the $\overline{AUPRC_w}$, we calculate the $AUPRC$ of each class independently and combine the individual values by calculating an average over all classes weighted by the number of instances that belongs to each class, that is,

$$\overline{AUPRC_w} \equiv \frac{\sum_i AUPRC_i \times S_i}{\sum_i S_i}; \tag{6.4}$$

where S_i is the number of instances in the i-th class.

6.3 Predictive Performance Results

In this section we show the predictive performance results for the algorithms we tested (the baseline algorithms PCT, PCTEN, LHC, ELHNB; our three Dependence Network (DN)-based algorithms HDN, HDN-PCT, and HDN-nHPC, and our hybrid PCT-LHC algorithm) across the 42 hierarchical datasets we have created or collected. Tables 6.5, 6.6 and 6.7 show the predictive performance results for each one of the measures we used: $AU(\overline{PRC})$, $\overline{AUPRC_w}$ and $\overline{AUPRC}$ respectively.

Next, for convenience, we briefly describe the hierarchical classification algorithms that we are evaluating:

1. The Predictive Clustering Tree (PCT) algorithm builds a decision tree by recursively splitting the dataset into clusters, using the value of the features (for more details see Section 2.2.2).

2. The PCT Ensemble (PCTEN) algorithm builds an ensemble of PCTs classifiers induced using bagging (for more details see Section 2.2.2).

3. The Hierarchical Dependence Network (HDN) algorithm is the first version of our DN algorithm for hierarchical classification (for more details see Section 5.2).

4. The hybrid HDN-PCT algorithm is the second version of our DN algorithm for hierarchical classification, in which the HDN algorithm is run in PCT's clusters (leaf nodes) with more instances than a pre-established value (for more details see Section 5.3).

5. The HDN based on finding non-Hierarchically related Predictive Classes algorithm (HDN-nHPC) is the third version of our DN algorithm, using a different approach to build and use the Markov blanket of the class nodes (for details see Section 5.4).

6. The LHC algorithm consists in inducing a local, standard ('flat') classification model to predict each one of the hierarchical classes and then combining the local model's predictions (for more details see Section 2.2.1).

7. The hybrid PCT-LHC algorithm is similar to the HDN-PCT algorithm, using the Local Hierarchical Classification (LHC) algorithm instead of HDN in the PCT's clusters (leaf nodes) with more instances than a pre-defined value (for details, see Section 5.5).

8. Finally, the Extended Local Hierarchical Naive Bayes algorithm (ELHNB) is a version of the Naive Bayes algorithm designed to work in the hierarchical classification setting, explained in Section 4.2.

We have applied the non-parametric *Friedman test* to our results. The Friedman test is designed to detect differences in classifiers' performances when they are applied to multiple datasets. This test, however, does not tell us which classifiers are statistically significantly different from other classifiers and which classifiers are statistically equivalent, only that there are differences in classifiers' performances (Demsar 2006).

To get that more specific information, we use the *Nemenyi test*, which is a post-hoc non-parametric statistical test that considers all possible pairs of classifiers and informs us which pairs of classifiers have significantly different performance, if the Friedman test has detected significant differences in classification performance (Demsar 2006).

The *Friedman test* has detected statistically significant differences when considering all three measures ($AU(\overline{PRC})$, $\overline{AUPRC_w}$ and $\overline{AUPRC}$); therefore, we have applied the Nemenyi test to detect significant differences among pairs of classifiers. Figures 6.3, 6.4, and 6.5 show the results of the Nemenyi tests ($\alpha = 0.05$)

graphically using critical diagrams (Demsar 2006). These diagrams show, on the horizontal axis, the mean rank of each algorithm, and above the horizontal axis the critical distance (CD), i.e., the minimal distance that two ranks must have in order for the respective algorithms to be considered statistically significantly different.

In the lower part of each figure we show which algorithms are statistically significantly different from the others by connecting with a horizontal line algorithms that have statistically equivalent predictive performance. Algorithms that are not connected by a horizontal line have statistically significantly different predictive performances, and the ones with lowest ranks have superior performance.

Table 6.5 shows the predictive performance of the hierarchical classification algorithms considering the $AU(\overline{PRC})$ measure. This table shows that one of our algorithms, HDN-nHPC, had the best (smallest) overall average rank (2.5) across all datasets compared to the other hierarchical classification algorithms.

Note, however, that the performance of the HDN-nHPC hierarchical classification algorithm was not statistically significantly different than the other three ("PCTEN","LHC","PCT-LHC") of the top four hierarchical classification algorithms we have tested, according to the Nemenyi statistical test, as shown in Figure 6.3.

It is interesting to note that, for the $AU(\overline{PRC})$ measure and the datasets we have considered, the LHC algorithm was statistically significantly better than the well-known PCT algorithm. It is surprising that a collection of local classifiers had better performance than the PCT algorithm, which is specially tailored to classifying hierarchical data.

By analysing Table 6.5 in more detail we can see that the HDN-nHPC, PCTEN, and LHC were always among the top three algorithms in terms of average rank in each one of the four major dataset types we have considered (Vens' GO, Vens' FunCat, Ageing GO, and Ageing MPO). It is interesting to note that although the HDN-nHPC algorithm was not the best algorithm in any of the four major dataset types, it had the best overall algorithm rank. It appears that the PCTEN algorithm did not perform so well on the Venn's datasets, specially when using the FunCat hierarchy. On the other hand, the LHC algorithm did not perform so well on the Ageing datasets.

Other interesting results include the fact that the PCTEN algorithm clearly outperformed the PCT algorithm in all four major dataset types, and that the hybrid HDN-PCT always had at least the same (and often better) average rank in comparison with the HDN algorithm in all four major dataset types. In addition, the hybrid HDN-PCT and the stand-alone HDN obtained overall ranks of 5.3 and 6.5, respectively. These results show that the hybridisation indeed improved the predictive performance of the HDN algorithm when considering the $AU(\overline{PRC})$ measure. However, the hybrid HDN-PCT was overall worse than the PCT algorithm (with overall rank of 4.7).

Similarly, the PCT-LHC hybrid (with overall ranking of 4.1) performed better than PCT algorithm (with overall ranking of 4.7), indicating that the hybrid improved in relation with the PCT algorithm; however, the hybrid did not improve with relation to the LHC algorithm (with overall rank of 3.0). Actually, the same behaviour happened across the other two predictive measures ($\overline{AUPRC_w}$ and $\overline{AUPRC}$), the hybrids PCT-LHC and HDN-PCT always outperformed the overall worst classifier of the pair (HDN for HDN-PCT and PCT for PCT-LHC) but did not outperform the best classifier (PCT for HDN-PCT and LHC for PCT-LHC). Therefore, we can conclude that the hybrids were not successful overall across the datasets we have used, as one of the two classifiers composing the hybrid classifier always had at least equivalent performance in relation to the hybrid classifier.

Comparing the rank results in Table 6.5 and Figure 6.3 (for the $AU(\overline{PRC})$ measure) with the ones in Table 6.6 and Figure 6.4 (for the $\overline{AUPRC_w}$ measure), we can see that the ordering of the algorithms in terms of their ranks is very similar for the measures $AU(\overline{PRC})$ and $\overline{AUPRC_w}$, with the exception of the HDN-nHPC and PCTEN algorithms (the top two algorithms, overall), which have an inverted overall average rank order in Tables 6.5 and 6.6, as well as in the corresponding Figures 6.3 and 6.4.

Interestingly, although the overall rank order for the algorithms in Tables 6.5 and 6.6 were very similar, the ranks across the four major dataset types were significantly different. In fact, in Table 6.6, the HDN-nHPC algorithm had the best predictive performance on both Venn's major datasets types but was among the three worst algorithms when considering the other two major datasets types. The opposite behaviour can be observed when analysing the ELHNB classifier: it performed very badly on the Venn's major dataset types but had good performance on the other two datasets types.

These performance differences come from the different biases that each predictive performance measure has: generally speaking, the $AU(\overline{PRC})$ measure favours more conservative classification models (models with lower complexity that focus more on classifying more generic classes), while the $\overline{AUPRC}$ measure favours more risk-prone models (predicting more specific classes), and the $\overline{AUPRC_w}$ measure falls in the middle of the risk proneness, favouring moderately risk-prone models. A clear evidence of this is that the PCT models induced to maximise the $AU(\overline{PRC})$ measure are always as shallow or shallower than the PCT models induced to maximize the $\overline{AUPRC}$ measure, and the depth of the PCT models induced to maximise the $\overline{AUPRC_w}$ measure tend to have intermediary depth (Vens et al. 2008).

For this reason, it is expected that the models induced to maximise the measures $\overline{AUPRC_w}$ and $\overline{AUPRC}$ will have higher variance, since these models predict more specific classes (at deeper class levels) more often than the models induced to maximise the $AU(\overline{PRC})$ predictive performance measure; and these predictions, which are based on fewer positive instances, have naturally higher predictive performance variance.

For the results based on the $\overline{AUPRC}$ measure presented in Table 6.7 and Figure 6.5, we can see that the algorithm's ranks were considerably different from the ones for the measures $AU(\overline{PRC})$ and $\overline{AUPRC_w}$. In particular, contrary to the other two measures, for the $\overline{AUPRC}$ measure, the PCTEN algorithm had worse predictive performance results than the PCT algorithm. Also, our HDN-nHPC algorithm, which had very good predictive performance results when using the other two measures, performed poorly (being the last in terms of overall average ranking).

Note that the very low accuracy values in Table 6.7 is due to the fact that the $\overline{AUPRC}$ measure averages the AUPRC results of several classes, without adjusting for the number of instances in each class. Because the majority of classes tend to annotate very few instances, the AUPRC measure for those classes tend to be low due to the increased difficulty of inducing a classification model with few positive instances. This kind of result can also be observed in other works dealing with hierarchical classification that used some of the datasets we have used for testing our algorithms (Vens et al. 2008; Cerri et al. 2016; Pugelj and Dzeroski 2011).

Another algorithm that we proposed, the PCT-LHC algorithm, was tied with the LHC classifier as the best performing algorithm in Table 6.7, but these two algorithms were the fourth and third best ones, respectively, in Tables 6.5 and 6.6. We attribute these differences of performance for these two algorithms to the fact that the $\overline{AUPRC}$ measure does not assume that classification errors close to the root of the hierarchy should be considered more serious than errors further away from the root of the hierarchy. Therefore, note that algorithms that focus more on improving the performance on specific classes (like the LHC algorithm), tend to perform better when considering the $\overline{AUPRC}$ measure. In other words, the good performance of both LHC-PCT and LHC regarding this measure can be explained by the fact that both these algorithms cope better with deeper classes than PCT, because they both apply LHC to leaf nodes, and LHC focuses on predicting each class separately. By contrast, PCT always tries to find good splitting conditions for several classes at the same time, which is hardly effective for deeper classes with low numbers of instances.

Note that, although the hybrid PCT-LHC and the stand-alone LHC were tied in terms of average rank, the hybrid PCT-LHC has an advantage in terms of model interpretability, since it builds a hierarchical classification model that is partly global. That is, part of its model consists of the decision tree built by the PCT algorithm, and that decision tree can be interpreted as a global model, which is easier to interpret than large set of local models built the LHC algorithm.

After analysing Table 6.7 (for the $\overline{AUPRC}$ measure) in more detail, we can see that the ranking results for datasets types Ageing GO, Ageing MPO and Venn's FunCat were broadly similar to the ranking results for the $\overline{AUPRC_w}$ measure (Table 6.6). The main differences concentrated in the Venn's GO datasets types, this is due to the fact that these datasets have more and deeper GO terms than the other datasets. For this reason, deeper classes with relatively fewer positive annotations influence more the results than in other datasets. Because these classes naturally have higher performance variance, it is natural that the results for the $\overline{AUPRC}$ measure in the Venn's GO datasets are more different than the results for the $\overline{AUPRC_w}$ and $\overline{AUPRC}$ measures in the other datasets.

Table 6.5: Predictive performance (%) results for the $AU(\overline{PRC})$ measure

Class Hier.	Feat. Type	PCT	PCTEN	HDN-PCT	HDN	PCT-LHC	LHC	HDN-nHPC	ELHNB
Vens' GO Datasets	seq	47.6	49.9	47.8	45.2	48.0	49.6	49.9	43.3
	cellcycle	44.4	46.4	44.4	42.4	44.6	47.2	47.3	43.2
	church	44.3	44.9	44.3	41.1	44.4	44.6	44.6	43.3
	derisi	44.9	45.2	44.8	41.3	44.9	45.1	45.2	43.2
	eisen	46.1	49.3	46.0	45.4	46.3	49.9	50.0	44.5
	gasch1	45.4	48.2	45.5	43.9	45.6	49.0	49.2	43.3
	gasch2	45.1	46.8	45.0	42.8	45.4	47.8	47.8	43.3
	spo	45.0	45.2	44.9	41.3	45.1	45.5	45.5	43.2
	expr	45.0	48.5	45.0	44.0	45.1	48.6	48.5	43.3
	struc	45.3	43.8	46.3	41.0	45.2	45.2	43.2	43.3
	hom	48.7	43.4	48.7	43.4	48.6	47.3	43.7	43.3
	Avg. Rank	4.7	3.0	4.8	7.5	3.9	2.4	2.4	7.4
Vens' FunCat Datasets	seq	21.5	24.4	23.1	25.6	22.5	25.9	25.7	14.9
	cellcycle	17.4	20.5	18.1	21.4	18.3	21.9	21.8	15.5
	church	17.3	17.7	17.1	16.9	17.3	17.4	17.4	15.5
	derisi	18.0	18.7	18.6	18.2	18.9	18.9	18.9	15.6
	eisen	20.1	24.9	20.9	25.1	20.7	25.4	25.4	15.8
	gasch1	20.6	24.2	22.7	23.7	22.8	24.1	24.1	15.5
	gasch2	19.3	21.4	19.6	22.1	19.6	22.9	22.7	15.4
	spo	19.0	19.7	19.7	19.0	20.0	19.3	19.3	15.6
	expr	20.9	24.9	22.5	23.6	22.3	24.0	24.0	15.4
	struc	18.4	15.3	19.4	18.0	19.0	17.9	14.8	15.2
	hom	25.8	15.0	25.9	20.7	25.9	20.8	15.9	15.0
	Avg. Rank	6.0	3.5	4.4	4.4	4.0	2.5	3.2	7.9
Ageing GO	worm Numeric	41.1	47.4	39.1	39.2	41.7	41.8	43.1	41.1
	PPI	41.3	43.2	39.5	38.4	40.6	41.1	42.0	31.5
	Motifs	48.5	48.1	47.7	40.6	48.8	43.3	44.5	38.9
	fly Numeric	42.1	48.1	41.7	43.6	41.8	43.7	44.1	53.2
	PPI	49.2	53.8	49.2	43.9	49.2	44.2	44.1	44.5
	Motifs	45.8	49.3	45.8	43.8	45.8	44.0	44.1	39.1
	human Numeric	45.5	45.5	41.3	41.0	45.7	46.5	46.9	44.5
	PPI	47.3	50.9	43.7	44.0	45.1	46.2	50.7	30.2
	Motifs	47.2	48.7	43.1	42.3	46.9	48.6	49.6	40.5
	mouse Numeric	46.6	46.6	44.5	44.5	46.3	46.4	47.1	43.8
	PPI	45.6	47.3	44.9	45.2	46.0	46.6	47.4	41.4
	Motifs	46.6	47.1	44.8	44.7	45.9	46.5	46.9	40.7
	yeast Numeric	42.6	46.1	38.4	34.7	43.6	46.0	46.3	38.2
	PPI	44.9	52.7	41.6	43.3	45.0	47.7	47.4	37.6
	Motifs	42.4	43.9	38.5	32.1	43.1	44.3	45.3	38.2
	Avg. Rank	4.0	1.9	6.0	6.9	4.1	3.7	2.5	6.9
Ageing MPO	KEGG	71.5	72.2	70.0	67.9	71.6	71.6	71.7	67.9
	KEGGI	70.6	71.3	67.1	66.5	70.6	70.9	71.0	68.3
	Motifs	71.0	71.4	69.1	67.9	70.9	71.0	71.1	66.2
	Numeric	71.1	71.5	69.5	68.2	71.5	71.8	71.8	64.5
	PPI	70.9	70.9	67.4	67.3	70.8	70.9	71.0	69.2
	Avg. Rank	4.4	1.6	6.4	7.5	4.1	3.1	1.8	7.1
	Overall Avg. Rank	4.7	2.6	5.3	6.5	4.1	3.0	2.5	7.3

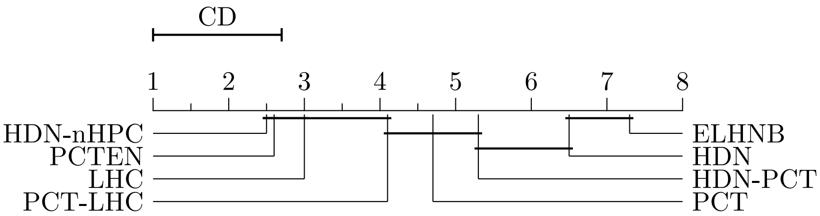


Figure 6.3: Comparison of all classifiers against each other using the Nemenyi test considering the $AU(\overline{PRC})$ measure

Table 6.6: Predictive performance (%) results for the $\overline{AUPRC_w}$ measure

Class Hier.	Feat. Type	PCT	PCTEN	HDN-PCT	HDN	PCT-LHC	LHC	HDN-nHPC	ELHNB
Vens' GO Datasets	seq	35.6	38.8	35.5	36.3	35.5	38.2	38.9	28.7
	cellcycle	32.0	35.4	32.1	33.2	32.2	35.5	35.9	28.7
	church	30.9	31.7	31.0	30.0	31.0	31.1	31.1	28.7
	derisi	31.5	32.3	31.4	30.5	31.5	32.2	32.3	28.6
	eisen	34.6	38.2	34.6	36.1	34.6	38.3	38.5	29.6
	gasch1	33.8	37.3	34.2	34.8	34.4	37.8	38.2	28.7
	gasch2	33.0	35.1	33.0	33.8	33.1	36.3	36.7	28.7
	spo	31.8	31.9	31.9	31.3	31.8	32.9	33.1	28.6
	expr	33.7	37.8	34.0	34.8	34.2	37.3	37.7	28.7
	struc	32.1	29.9	32.1	30.8	32.1	31.4	29.8	28.8
	hom	36.7	29.8	36.6	33.6	36.6	34.4	30.3	28.7
	Avg. Rank	5.2	3.1	5.0	5.0	4.7	2.7	2.3	8.0
Vens' FunCat Datasets	seq	16.9	20.4	16.9	20.6	16.9	20.9	21.4	9.9
	cellcycle	13.6	17.1	14.0	17.5	13.9	17.7	17.8	10.2
	church	11.8	12.7	12.3	12.7	12.1	12.4	12.4	10.2
	derisi	13.2	14.4	13.2	13.9	13.2	14.4	14.4	10.3
	eisen	15.0	20.2	15.1	19.9	15.1	20.1	20.4	10.5
	gasch1	16.4	19.7	16.4	19.4	16.4	19.5	19.9	10.2
	gasch2	14.1	17.0	14.1	17.5	14.2	18.1	18.3	10.2
	spo	14.7	15.4	14.9	15.2	14.7	15.2	15.4	10.3
	expr	16.0	20.5	16.1	19.8	16.1	19.9	20.1	10.2
	struc	15.5	11.7	15.5	13.8	15.5	13.9	11.9	10.1
	hom	22.2	12.0	22.2	17.5	22.2	17.2	12.1	9.9
	Avg. Rank	5.7	3.3	5.0	3.5	5.1	3.0	2.4	8.0
Ageing GO	worm Numeric	31.9	33.5	31.8	30.2	31.8	31.0	31.7	32.3
	PPI	33.4	30.8	31.2	30.0	33.4	32.2	30.7	32.5
	Motifs	34.6	33.1	33.9	31.8	34.7	33.0	33.0	34.7
	fly Numeric	39.8	38.3	39.8	38.4	39.8	38.4	38.6	38.6
	PPI	40.9	40.9	40.9	38.5	40.9	38.4	38.4	40.5
	Motifs	40.4	40.2	40.4	38.3	40.4	38.6	38.4	40.7
	human Numeric	36.0	35.0	35.9	33.2	36.0	34.3	35.0	32.9
	PPI	38.2	39.6	37.6	37.8	38.0	39.1	39.4	38.8
	Motifs	38.6	37.9	36.9	35.6	39.2	38.6	38.8	40.2
	mouse Numeric	37.6	36.7	37.6	36.2	37.6	36.2	36.8	37.2
	PPI	39.2	38.5	38.6	37.1	39.6	38.4	37.0	40.3
	Motifs	37.8	37.7	37.9	36.6	38.2	37.6	36.9	39.9
	yeast Numeric	32.2	34.1	31.9	30.0	32.1	33.8	34.7	32.1
	PPI	36.6	39.4	36.0	37.9	36.0	38.3	37.0	37.7
	Motifs	31.6	31.9	30.6	30.2	32.1	32.5	33.4	35.6
	Avg. Rank	3.4	4.2	4.6	7.2	3.2	5.2	5.1	3.1
Ageing MP0	KEGG	55.6	56.3	55.0	53.3	55.6	55.1	54.6	56.0
	KEGGI	54.7	54.6	53.7	54.0	54.1	53.6	52.6	55.9
	Motifs	54.5	55.1	54.2	52.7	54.6	53.9	53.6	54.4
	Numeric	53.7	54.6	53.9	54.5	54.0	55.1	55.2	53.6
	PPI	54.4	54.1	52.9	52.5	54.1	54.1	52.9	55.1
	Avg. Rank	3.5	2.4	5.9	6.6	3.7	4.8	5.9	3.2
	Overall Avg. Rank	4.5	3.5	5.0	5.6	4.1	3.9	3.7	5.7

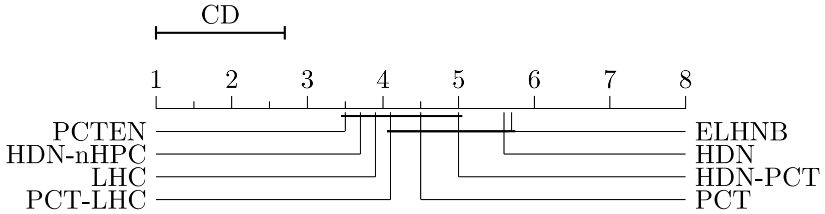


Figure 6.4: Comparison of all classifiers against each other using the Nemenyi test considering the $\overline{AUPRC_w}$ measure

Table 6.7: Predictive performance (%) results for the $\overline{AUPRC}$ measure

Class Hier.	Feat. Type	PCT	PCTEN	HDN-PCT	HDN	PCT-LHC	LHC	HDN-nHPC	ELHNB
Vens' GO Datasets	seq	2.3	2.0	2.3	2.3	2.3	2.6	1.9	0.9
	cellcycle	1.7	1.6	1.7	1.8	1.7	2.1	1.4	0.9
	church	1.5	1.2	1.5	1.4	1.5	1.5	1.0	0.9
	derisi	1.6	1.3	1.6	1.5	1.6	1.7	1.2	0.9
	eisen	2.6	2.4	2.6	2.4	2.6	2.7	1.9	1.1
	gasch1	2.1	2.0	2.1	2.0	2.1	2.4	1.7	0.9
	gasch2	1.8	1.6	1.8	1.8	1.8	2.1	1.5	0.9
	spo	1.9	1.5	1.9	1.6	1.9	1.8	1.2	0.9
	expr	1.9	2.0	1.9	2.0	1.9	2.3	1.6	0.7
	struc	2.0	0.9	2.0	1.5	2.0	1.6	0.9	0.9
	hom	3.3	0.9	3.3	2.1	3.3	2.2	1.0	0.9
	Avg. Rank	3.0	5.8	3.0	4.3	3.0	2.0	6.9	7.9
Vens' FunCat Datasets	seq	4.2	4.8	4.2	4.7	4.2	4.7	4.3	1.7
	cellcycle	2.9	3.5	3.0	3.8	3.0	3.9	3.4	1.8
	church	2.4	2.5	2.5	2.6	2.4	2.5	2.2	1.8
	derisi	2.9	2.9	2.9	2.9	2.9	3.0	2.7	1.8
	eisen	3.7	4.8	3.7	4.7	3.7	4.8	4.1	2.1
	gasch1	3.8	4.4	3.8	4.4	3.8	4.5	4.0	1.8
	gasch2	2.9	3.6	2.9	4.0	2.9	4.1	3.6	1.8
	spo	3.1	3.0	3.1	3.1	3.1	3.1	2.8	1.8
	expr	3.7	4.8	3.7	4.5	3.7	4.6	4.0	1.8
	struc	3.1	2.1	3.1	2.7	3.1	2.8	2.0	1.8
	hom	5.9	2.1	5.9	3.8	5.6	3.7	2.2	1.7
	Avg. Rank	4.8	3.5	4.5	2.9	4.8	2.3	5.2	8.0
Ageing GO	worm Numeric	9.6	10.1	9.6	8.5	9.6	8.7	8.9	9.4
	PPI	9.8	8.8	9.5	8.4	9.8	9.0	8.6	9.6
	Motifs	10.5	9.8	10.4	8.9	10.6	9.2	9.2	10.5
	fly Numeric	13.3	12.8	13.3	12.8	13.3	12.8	12.8	12.8
	PPI	13.5	13.6	13.5	12.8	13.5	12.8	12.8	13.6
	Motifs	13.4	13.3	13.4	12.8	13.4	12.8	12.8	13.5
	human Numeric	9.4	9.1	9.4	8.5	9.4	8.6	8.8	8.4
	PPI	10.0	10.3	9.9	9.6	10.0	9.9	10.0	10.6
	Motifs	10.3	9.8	10.0	9.2	10.5	10.0	9.8	11.0
	mouse Numeric	13.2	12.5	13.2	12.6	13.2	12.6	12.7	12.8
	PPI	13.8	13.2	13.7	12.8	13.9	13.1	12.7	14.1
	Motifs	13.3	13.1	13.2	12.7	13.4	12.9	12.8	14.0
	yeast Numeric	8.0	8.5	8.0	6.8	8.0	8.1	8.3	7.9
	PPI	9.6	11.0	9.6	9.7	9.6	9.9	12.7	11.4
	Motifs	7.9	7.9	7.8	7.2	8.0	8.4	8.1	10.5
	Avg. Rank	3.4	4.3	4.1	7.3	3.0	5.5	5.4	3.0
Ageing MP0	KEGG	14.6	14.8	14.3	13.3	14.6	14.1	13.8	15.0
	KEGGI	14.1	14.0	13.5	13.9	13.7	14.1	12.9	15.2
	Motifs	13.6	13.9	13.4	12.8	13.6	13.4	13.1	13.9
	Numeric	12.8	12.9	12.8	13.2	12.9	13.4	13.4	12.7
	PPI	13.6	13.2	12.6	12.3	13.3	13.0	12.4	13.8
	Avg. Rank	3.6	3.3	6.0	6.4	4.1	4.1	6.1	2.4
	Overall Avg. Rank	3.7	4.4	4.2	5.3	3.6	3.6	5.8	5.5

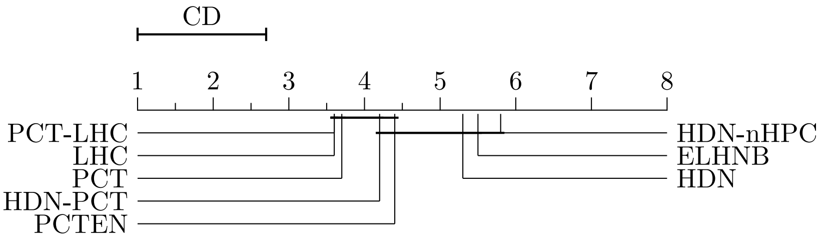


Figure 6.5: Comparison of all classifiers against each other using the Nemenyi test considering the $\overline{AUPRC}$ measure

6.4 Model Interpretation Results

In this section we shall interpret PCT models to extract potentially relevant ageing-related knowledge from them. It is worth noticing that interpreting the PCT models is valuable for the stand-alone PCT models and the hybrid algorithms that use the same PCT model as a base to induce further classification models (HDN-PCT and PCT-LHC).

We focus on interpreting PCT models, rather than DN models, because the former are simpler (smaller) global models, i.e., each PCT tree contains predictions for all hierarchical classes; whereas each DN-based model contains a large collection of local classification models, i.e., each local model contains predictions for a single class.

The interpretation of the hierarchical classification models was performed in conjunction with Dr. Jennifer M. A. Tullet, an expert on the biology of ageing, and published in (Fabris, Freitas and Tullet 2015).

6.4.1 Coverage Scores of Features in the PCT Models

We calculate the coverage score of a given feature by building a PCT decision tree (model) using all available instances in the entire dataset and dividing the number of instances that used that feature for their classification (i.e., the feature occurs in the path from the root to the leaf node where the instance is classified) by the total number of instances that were classified. Higher coverage values correspond to more “useful” features, i.e., features that were used more often to predict which class labels are associated with each instance (gene or protein). In particular, a feature in the root node has the maximum coverage of 1, since that feature is used to classify all instances in the dataset; whereas features in deeper tree nodes have lower coverage, since they are used to classify fewer instances.

Table 6.8 presents the features with a coverage score greater than 0.5 in the PCT decision trees for the dataset with ageing-related GO terms as classes and PPI features using the s-value that maximises the $\overline{AUPRC}$ measure. We have chosen this measure because it generates larger decision trees, maximising the opportunity of finding interesting relations between features and ageing-related GO terms. We focus on the Ageing GO PPI dataset because the PPI features have greater interpretability compared to the numeric and motif features, the GO classes are very popular among biologists, and the GO hierarchy annotates a more diverse set of animal species.

Table 6.8 shows for each feature: its rank, from most relevant (1) to least relevant; the feature identifier; the full name of the feature; and the feature's coverage score.

Analysing the human dataset, the top-ranked PPI feature for predicting ageing-related GO terms is represented by interaction with CREBBP (CREB binding protein), an acetyl transferase involved in the acetylation of histones and other proteins within the cell (Bedford and Brindle 2012). CREBBP has diverse functions and is important for development, physiology and disease. One of its targets is the FOXO transcription factor (Daitoku et al. 2004), a protein tightly linked to longevity and ageing in several species, including humans (Guevara-Aguirre et al. 2011).

Mutations in the gene that encodes the protein in the second ranked interaction, PTPN11 (protein tyrosine phosphatase, non-receptor type 11), are known to cause myeloid leukemia, dramatically reducing human life expectancy (Hou et al. 2007). In particular, activating mutations in this gene have been shown in cell culture to cause proliferative arrest and senescence. The mechanisms of which could provide insight into the initial onset of PTPN11-related cancers (Zheng et al. 2013).

The third-ranked feature in the human dataset represents interaction with TP53 (Transformation-related Protein 53), commonly known as p53 and an important tumor suppressor that has been described as "the guardian of the genome". Its relationship with ageing is complex as decreased expression of the gene leads to tumor formation which increases mortality, but in contrast certain mutations in the gene have also been linked to increased life expectancy in humans (van Heemst et al. 2005). Interestingly, interaction with TP53 was also selected as an important feature in the mouse dataset. This supports the hypothesis that this protein may play an important role in the prediction of ageing-related GO terms.

For the yeast dataset, the top 13 features illustrate a wide variety of molecular interactions, making it difficult to comment on individual processes. However, one of the top genes in the list is Ribosomal protein 51. Ribosomal proteins, via their role in translation, are strongly implicated in lifespan regulation in several species and represent a key mode of ageing modulation (Charmpilas et al. 2014).

6.4.2 Interpreting Classification Models

In this section we present some classification models (decision trees) generated by the PCT algorithm and the analysis of some over-represented ageing-related GO

Table 6.8: PPI features with coverage score > 0.5 in the decision trees built by the PCT algorithm for the $\overline{AUPRC}$ measure and Ageing GO classes.

Organism	Rank	Feat. Id.	Full Name	Score
Worm	1	LET60	LEThal family member	1.00
	2	wei	Molecular weight	0.94
Fly	1	AMN	Amnesiac	1.00
	2	len	Sequence length	0.92
Human	1	CREBBP	CREB binding prot.	1.00
	2	PTPN11	Tyrosine-prot. phosphatase non-rec. type 11	0.83
	3	TP53	Transformation-related Prot. 53	0.76
	4	VTN	Vitronectin	0.52
	5	NOTCH1	Notch homolog 1	0.51
Mouse	1	TP53	Transformation related prot. 53	1.00
	2	POU5F1	POU domain, class 5, transcription factor 1	0.92
Yeast	1	HHT1	Histone H3	1.00
	2	RPS17B	Ribosomal prot. 51	0.85
	3	ATP6	ATP synthase	0.80
	4	PIF1	PIF1 5'-To-3' DNA Helicase	0.79
	5	FCY2	Purine-cytosine permease	0.77
	6	CYR1	Adenylate cyclase	0.76
	7	HOS2	Histone deacetylase and subunit of Set3 and Rpd3L complexes	0.73
	8	VPS38	Vacuolar prot. sorting-assoc. prot. 38	0.70
	9	len	Molecular length	0.67
	10	ATG12	Ubiquitin-like prot. ATG12	0.67
	11	IDH2	Isocitrate dehydrogenase (NAD) subunit 2, mitochondrial	0.56
	12	BAS1	Myb-like DNA-binding prot. BAS1	0.52
	13	CLN1	G1/S-specific cyclin CLN1	0.51

terms in some leaf nodes of the decision tree. Recall that the PCT algorithm generates a decision tree that contains, in each node, a test that splits the testing instances into two different paths. When an instance reaches a leaf node (a cluster), a class (GO term) probability vector is assigned to it.

We focus again on the PPI features for predicting ageing-related GO terms, which are easier to interpret in comparison with the numeric and motif features, and we discard the human and yeast decision trees (with 13 and 52 nodes, respectively), because their larger sizes make interpretation more difficult.

Figures 6.6 to 6.8 show some over-expressed GO terms and the decision tree model generated by the PCT algorithm maximising the $\overline{AUPRC}$ measure for the worm, fly and mouse organisms, respectively. The choice of over-expressed GO terms in the tables in Figures 6.6b, 6.7b and 6.8b took into account both their small p-value and their relevance to ageing based on current biological knowledge.

In the models, the first number in the pair before the first decision split represents the *a priori* mean entropy of the class labels. Entropy is defined as: $H(\hat{P}) = -\sum_i \hat{P}_i \log(\hat{P}_i)$, where $\hat{P}_i$ is the proportion of instances with class i in the current node of the decision tree. Smaller entropy values represent sets of instances that have more separated (more reliably predicted) classes. The second number in the pair before the first decision split represents the total number of instances classified by the decision tree. In the leaf nodes, the pair of numbers represents the entropy and number of instances in the current leaf node.

In the decision tree shown in Figure 6.6a, if a given protein does not interact with “LET60” (LEThal family member 60), then the protein is assigned to a cluster based on its molecular weight. In Figure 6.6a the number of proteins that do not interact with “LET60” is 246, almost all of the 263 instances, and the entropy value did not decrease significantly. Hence, *not* interacting with the “LET60” protein is not a good predictor of ageing-related GO terms. On the other hand, proteins that interact with both “LET60” and “LIN45” (assigned to cluster 3) have a much smaller class entropy compared to other clusters, indicating that these two protein interactions are relevant predictors of ageing-related GO terms.

The table in Figure 6.6b, below the decision tree, shows 3 over-expressed ageing-related GO terms (class labels) in the clusters of the two leaf nodes for the worm Ageing GO PPI dataset. This table shows: the cluster ID, the GO term id and name, and the probability that the number of occurrences of the GO term in the current cluster is greater than or equal to the number of observed occurrences, assuming that the occurrence of the GO term in an instance follows a Bernoulli distribution with the GO term's frequency in the training dataset used as the 'success' probability. The table also displays, before the GO terms of each cluster, the feature-based conditions in the decision tree that must be satisfied in order for an instance to be assigned to a particular cluster.

Considering the worm dataset, of the GO terms reaching a significance of 10^{-4} or less, almost half are related to developmental processes. Perhaps this is not surprising as development and, more recently, growth, have been implicated in ageing related processes (Kenyon 2010; Gems and de la Guardia 2013). The remaining significant GO terms were divided almost equally between involvement in either reproductive processes or signalling pathways. Again, it is difficult to argue against the importance of either of these in ageing; particularly the latter, where the use of genetics and molecular biology have allowed many key ageing pathways to be dissected in worms (Kenyon 2010).

Figure 6.6b suggests that the feature representing interaction with the "LET60" (LEThal family member 60) protein is a good predictor for GO terms related to organism development in worms. This is consistent with the fact that the let-60 gene encodes the C. elegans' Ras protein, which is central to a variety of different signaling pathways. One of these is the RTK-Ras-ERK pathway, which is well conserved between species (Udell et al. 2011).

This pathway is critical during development and controls many biological processes during adulthood. Mutations in components of the RAS pathway are also implicated in many human syndromes and diseases, e.g. cancer (Karnoub and Weinberg 2008; Tidyman and Rauen 2009).

In worms, let-60 is expressed in neural, muscle, and hypodermal lineages and its activity is required for proper larval development. It has also been linked to ageing and shown to promote longevity. Two suggested mechanisms for this are: 1) Promoting protein homeostasis via the ubiquitin proteosome system (UPS) (Liu et al. 2011) and; 2) Activation of the transcription factor SKN-1, which represses insulin-like peptide expression and down regulates the insulin signalling pathway (Okuyama et al. 2010). There are also other, less direct, implications that Ras acts to control ageing and healthspan. Thus, it is interesting that (based on the large coverage score) interaction with LET60 is the most important discriminator for ageing-related GO terms in the worm Ageing GO PPI dataset.

Figure 6.7a shows the decision tree generated for the fly dataset, where interactions with the AMN (Amnesiac) protein greatly reduce the entropy of the class labels.

```
(Ent.: 77.50, #Inst.: 263)
LET60 = No (74.92, 246)
|     wei > 96753.69 (88.06, 53)
|          Cluster 0
|     wei <= 96753.69 (68.51, 193)
|          Cluster 1
LET60 = Yes (69.41 17)
|     LIN45 = No    (66.77, 12)
|          Cluster 2
|     LIN45 = Yes (34.77, 5)
|           Cluster 3
```

(a) PCT classification model for the worm Ageing GO PPI dataset.

Clust.	GO term	p-value
	IF LET60 = Yes AND LIN45 = No	
2	GO:0050793 (Reg. of dev. proc.)	10^{-4}
	IF LET60 = Yes AND LIN45 = Yes	
3	GO:0022414 (Reproductive process)	10^{-5}
	GO:0023052 (Signalling)	10^{-5}

(b) Most statistically significant over-expressed ageing-related GO terms in the PCT model's leaf nodes for the worm Ageing GO PPI dataset.

Figure 6.6: PCT model and over-expressed GO terms (classes) in the worm Ageing GO PPI dataset

In the fly data set, most over-represented GO terms were involved in development, food recognition and behaviour, learning and memory. As with the worm, development and growth are not surprising. However, lifespan can be extended dramatically in a wide variety of organisms by reducing caloric intake (Mair and Dillin 2008), thus it is logical that feeding, and behaviour that influences this would affect lifespan.

Interestingly, as shown in Figure 6.7b, the over-expressed GO terms (class labels) in cluster 2 are associated with brain development. In fact, AMN has been shown to be required for normal brain development, sleep regulation and adult memory consolidation (Hashimoto, Shintani and Baba 2002). Both sleep regulation and memory decline with age in a number of species, and indeed AMN is linked to these processes in an age-dependent fashion in the flies (Tamura et al. 2003). Thus, it is interesting that this gene was identified as relevant by the PCT algorithm.

Figure 6.8a shows the decision tree built for the mouse dataset. We can see that the most important feature (the feature at the root of the decision tree) is

```
(115.88, 79)
AMN = No (112.60, 73)
|     len > 741.0 (127.81, 20)
|         Cluster 0
|     len <= 741.0 (95.58, 53)
|         Cluster 1
AMN = Yes (9.80, 6)
     Cluster 2
```

(a) PCT classification model for the fly Ageing GO PPI dataset.

Clust.	GO term	p-value
	IF AMN = Yes	
2	GO:0007631 (feeding behaviour)	10^{-6}
	GO:0007613 (memory)	10^{-6}
	GO:0048589 (dev. cell growth)	10^{-5}

(b) Most statistically significant over-expressed ageing-related GO terms in the PCT model's leaf nodes for the fly Ageing GO PPI dataset.

Figure 6.7: PCT model and over-expresed GO terms (classes) in the fly Ageing GO PPI dataset

TP53, which interestingly, was also selected by the model induced for the human organism.

There are fewer significant GO terms in the mouse data set and these are enriched for terms implicated in a variety of different signalling pathways, i.e. apoptotic pathways, cell cycle and regulation of gene expression. This contrasts with the worm and fly data where "developmental processes" predominate. However, it does complement the analysis based on feature coverage scores (see Table 6.8), showing that interaction with p53, PTPN11 and CREBPB (all key signalling molecules) is important for ageing.

6.4.3 Analysis of the Predictive Class Relationships

As an example of the benefit of detecting predictive class relationships and how to use them to improve predictive performance, Table 6.9 presents the 10 strongest predictive class relationships for the Numeric GO Ageing Yeast dataset. Relationships are ranked in decreasing order of the mean $AUPRC$ accuracy measure difference, calculated as the $AUPRC$ of the pair of classifiers ($M^{\tilde{c_i}}_{(i,i')}$ and $M^{c_i}_{(i,i')}$) minus the $AUPRC$ of the single classifier ($M_{(i,i')}$), over the 10 folds of the cross-validation procedure.

```
(190.68  107)
TP53 = No  (180.25,98)
|      POU5F1 = No  (178.78,92)
|           Cluster 0
|      POU5F1 = Yes  (96.07,6)
|           Cluster 1
TP53 = Yes  (190.97  9)
      Cluster 2
```

(a) PCT classification model for the mouse Ageing GO PPI dataset.

Clust.	GO term	p-value
	IF TP53 = Yes	
2	GO:051726 (regulation of cell cycle)	10^{-5}
	GO:008630 (intrinsic apoptotic signalling pathway in response to DNA damage)	10^{-4}
	GO:010468 (reg. of gene expression)	10^{-5}

(b) Most statistically significant over-expressed ageing-related GO terms in PCT model's leaf nodes for the mouse Ageing GO PPI dataset.

Figure 6.8: PCT model and over-expresed GO terms (classes) in the mouse Ageing GO PPI dataset

Table 6.9: Mean, maximum and minimum AUPRC differences (across the folds of the 10-fold cross validation process) between the pair-wise classifiers ($M^{\tilde{c_i}}_{(i,j)}$ and $M^{c_i}_{(i,j)}$) and the single classifier ($M_{(i,j)}$). From this table it is clear that the SVM classifier predicting class C_1 greatly benefits from the dataset split based on C_2 values.

C_1	C_2	Mean	Max.	Min.
GO:0046483	GO:0034641	0.56	0.50	0.59
GO:0034641	GO:0006725	0.55	0.51	0.62
GO:0006725	GO:0046483	0.54	0.51	0.58
GO:0006807	GO:0046483	0.53	0.52	0.53
GO:0034641	GO:1901360	0.51	0.41	0.58
GO:0006725	GO:0034641	0.51	0.49	0.54
GO:1901360	GO:0006807	0.49	0.43	0.60
GO:0046483	GO:0006807	0.49	0.46	0.53
GO:0034641	GO:0046483	0.49	0.35	0.57
GO:1901360	GO:0034641	0.47	0.42	0.52

It is clear that there are strong relationships in the yeast dataset that may be exploited by our algorithm. Table 6.10 presents additional information about the GO terms from Table 6.9. This table contains in its columns the GO term Id, its name, the average depth of the term and the average height of the term. The

Table 6.10: Information about the GO terms in Table 6.9

Go term	Name	Avg. Depth	Avg. Height
GO:0046483	heterocycle metabolic process	4.0	4.5
GO:0034641	cellular nitrogen compound metabolic process	4.0	4.0
GO:0006725	cellular aromatic compound metabolic process	4.0	4.8
GO:0006807	nitrogen compound metabolic process	3.0	4.8
GO:1901360	organic cyclic compound metabolic process	4.0	4.5

average depth of a term is computed by calculating the average length of all paths from the root node of the hierarchy to that GO term. Similarly, the average height of a GO term is computed by calculating the average length of all paths from that GO term to the reachable leaves of the term hierarchy.

The average depth and average height of a GO term inform us how specific that GO term is. Broadly speaking, deeper nodes tend to be more specific than shallower nodes; likewise, nodes with shorter heights tend to be more specific. More specific GO terms tend to provide more information to users of the classification system. Although average depth is a more common measure of term specificity, we also use the average height because sometimes a relatively shallower node may be more specific than a deeper one.

Table 6.10 shows that the selected GO terms are located, in general, in the middle of the GO hierarchy, as far as depth and height are concerned. In other words, there is a compromise between specificity and generality, probably because deeper GO terms are easier to differentiate from the other nodes but, at the same time, contain fewer labeled instances.

Figure 6.9 shows the ancestors of the GO terms present in Table 6.10, with solid edges representing the original GO relationships and dashed edges representing the predictive class relationships detected by the algorithm. It is clear that the relationships found by the algorithm connect classes that are semantically related; all of them are related to metabolism and are relatively close in the original class (GO) hierarchy. This is a sign that these predictive class relationships are in fact meaningful. Note that the class dependencies represented by the dashed edges (i.e., detected by our method) would be ignored by a conventional local hierarchical classification algorithm, which would consider only parent-child relationships in the original class hierarchy.

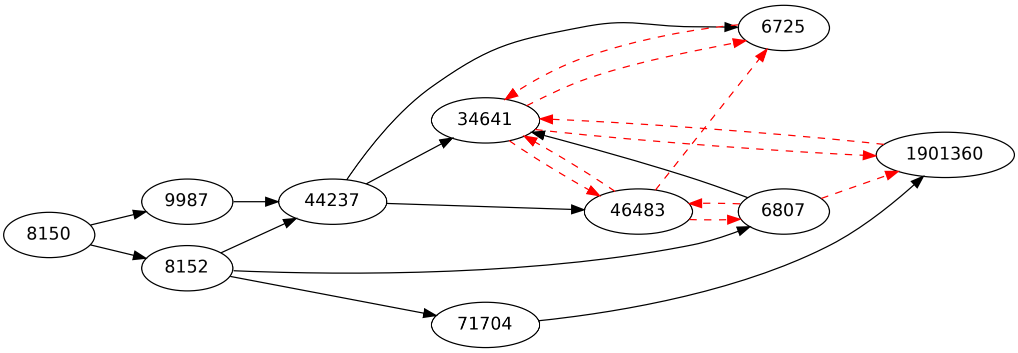


Figure 6.9: GO terms in Table 6.10 and their ancestors. Solid edges represent the original GO term relationships. Dashed edges represent the class dependencies present in Table 6.9.

6.5 Worst-Case Time Complexity Analysis of the HDN-nHPC Classification Algorithm

Unfortunately it is not possible to directly compare, in a fair way, the CPU clock runtime of all hierarchical classification algorithms in the datasets used in our experiments. This is due to two factors, as follows.

First, all the algorithms were run on a computer cluster with 60 processors, and when an algorithm is executed on that computer cluster, the number of processors actually allocated to run that algorithm depended on the number of processors available at that particular time, which varied depending on the number of users that were using the cluster at that time and on the runtime of their programs. Obviously, it is not fair to compare the CPU clock runtime of two or more algorithms run on different number of processors in the cluster.

Second, even if all algorithms were run on the same number of processors, in general, the algorithms were implemented in different and mixed combinations of programming languages, and the language(s) used has(ve) a significant effect on the runtime of an algorithm. To be more precise about the diversity of programming languages used, the algorithms can be divided into four groups, as follows: (a) the PCT and PCTEN algorithms were primarily implemented in the *Java* programming language, with a few integration parts implemented in the *Python* programming language; (b) the HDN, LHC and HDN-nHPC used a *C++* implementation of the "flat" SVM used as a base classification algorithm and *Python* for the rest of the algorithms; (c) the HDN-PCT and PCT-LHC hybrid algorithms involve hybrid *Python/C++/Java* implementations, with the PCT part in *Java*, the "flat" SVM in *C++*, and the HDN part and dataset preparations used by both HDN and LHC in *Python*; and (d) the ELHNB algorithm, which is fully implemented in *Python*.

Hence, instead of reporting CPU clock running times, in this section we perform a theoretical analysis of time complexity, which has the advantage of being programming language-independent. We focus on the worst-case time complexity of the HDN-nHPC algorithm, for two reasons. First, out of the new hierarchical classification algorithms proposed in the thesis, it was the most successful overall, in terms of predictive performance, as discussed earlier. Second, this algorithm was also overall the most time consuming hierarchical classification algorithm used in our experiments, mainly due to the procedure to find the non-hierarchical relationships and the multiple runs of an SVM algorithm used to predict each hierarchical class. Hence, a worst-case time complexity analysis of this algorithm will help us to understand the limitations of its scalability to very large datasets, as a function of several relevant variables describing the size of the classification problem or a parameter of the algorithm. More precisely, the variables used in our time-complexity analysis are as follows:

- N: number of classes.
- M: number of features.
- J: number of training/testing instances.
- it: number of Gibbs sampling iterations.

We present next the worst-case time complexity analysis of the training and testing phases of the HDN-nHPC hierarchical classification algorithm.

6.5.1 Training Time Complexity

We begin our analysis by the training phase of the algorithm, which is presented in Algorithm 5.2. Lines 2 and 3 are trivial, having negligible time complexity. The *for* loop starting at line 4 is executed, in the worst case, $(N-1)^2$ times if the class hierarchy is degenerated and has all hierarchical classes at the first level of the hierarchy, without edges between the nodes at the first level of the hierarchy. Therefore, the worst-case scenario complexity of the *for* loop starting at line 4 is $O(N^2)$.

Lines 5 and 6 have negligible time complexity, but at line 7 the algorithm runs the F-test feature selection algorithm, which has time complexity of $O(MJ)$, as calculating the F-statistic for each feature involves calculating the variance of the feature, which has linear complexity on the number of instances.

Next, in lines 8, 9 and 10, three SVM classification models are induced, by running the SVM algorithm three times. For our complexity analysis it is enough to consider only the SVM algorithm run at line 10, that uses the most instances. The training complexity of the SVM algorithm using the RBF kernel is $O(\max(J,M) \times \min(J,M)^2)$ (Chapelle 2007), which clearly dominates the complexity of running the F-test. Therefore, so far, the time complexity of Algorithm 5.2 is $O(N^2 \times \max(J,M) \times \min(J,M)^2)$.

Lines 11 to 18 iterate over each instance in the set D_{valid} (whose size is limited by $O(J)$) and get their prediction using the SVM algorithm. The complexity of the prediction phase of the SVM algorithm is $O(J^2)$ (Abdiansah and Wardoyo 2015). Therefore, the overall complexity of lines 11 to 18 is $O(J^3)$, and the overall algorithm complexity so far is $O(N^2 \times (\max(J,M) \times \min(J,M)^2 + O(J^3)))$.

Line 19 of the algorithm has linear time on the number of instances, therefore its time complexity is dominated by the other previously analysed lines. Line 20 is executed in constant time, therefore it is negligible. Line 23 takes $O(N^2)$ operations and is also negligible when compared to the complexity of the *for* loop starting at line 4.

The final time complexity of the training phase of the HDN-nHPC algorithm is, therefore, $O(N^2 \times (\max(J,M) \times \min(J,M)^2 + O(J^3)))$.

6.5.2 Testing Time Complexity

Now we analyse the time complexity of the testing phase of the HDN-nHPC algorithm, presented in Algorithm 5.4. This algorithm is called for every testing instance, therefore we begin with a complexity of $O(J)$. In line 2 the predictive class label vector is randomly initiated, which has a complexity of $O(N)$. This complexity, however, will be dominated by the following analysis and can be ignored.

Next, the *for* loop starting at line 3 is executed *it* times, bringing the current complexity to $O(J \times it)$. At line 4 the *for* loop iterates over all N class labels, bringing the complexity to $O(J \times it \times N)$.

Lines 5 is executed in constant time, hence it can be ignored. For our worst-case analysis, let us assume that the *if* condition at line 6 always fails. This assumption must be made because the time complexity of line 7 is always smaller than the time complexity of line 10, as line 10 will use at least one SVM classifier (and possibly more) to predict the class probability of the instance.

Now me must analyse line 10 of the algorithm. This line calls Algorithm 5.3, which has the worst-case complexity of $O(N \times J^2)$. This is the case if all other class labels are in the Markov blanket of every class label and the prediction of the SVM algorithm (which has the complexity of $O(J^2)$) must be retrieved $(N - 1)$ times.

This brings the overall complexity of the algorithm so far to $O(J^3 \times it \times N^2)$. Because the lines 11 to 19 of the algorithm have constant time, this is the final time complexity of Algorithm 5.4, which is the time complexity of the testing phase of the HDN-nHPC algorithm.

6.6 Conclusions

After analysing the results in Section 6.3, we have concluded that for one out of the three hierarchical predictive performance measures used in our experiments, one of our four new algorithms (the HDN-nHPC algorithm) outperforms all other seven algorithms in terms of average rank across the 42 hierarchical classification datasets.

In Section 6.4 we interpret the models generated by the PCT hierarchical classification algorithm. The interpretation of the PCT model has corroborated known biological facts about the biology of ageing and has also found protein-protein interactions in some model organisms that are known to be ageing related in other animals, which may indicate that the effect of the protein is conserved between species. We have also validated the predictive class relationships found by the HDN-nHPC algorithm, showing that the HDN-nHPC algorithm has found strong new relationships between hierarchical classes.

Chapter 7

Meta-Learning for Hierarchical Classification

7.1 Introduction

The cost of performing high-throughput biological assays has been constantly decreasing throughout the years. This increases the availability of freely available biological data and thus, the need for computational methods to help biologists extract useful knowledge from this data.

As we have previously presented, one of the computational tools available for biologists are *hierarchical classification algorithms*, these algorithms take as input a dataset containing instances described by numerical features and some class labels that annotate the instances, and learn a model that can be used to label unseen instances. Recall that, in hierarchical classification, the class labels are structured using ontologies structured as a tree or a Directed Acyclic Graph (DAG), where each node is a class label and each edge represents a "IS-A" relationship between labels. This means that if an instance is annotated with one class label, it is implicitly annotated with all of that label's ancestor labels.

Meta-learning for algorithm recommendation in classification problems is the computational task of recommending to the user a classification algorithm (or a ranking of classification algorithms) from a pool of algorithms, for any new classification dataset (new meta-instance), given the past performance of the algorithms in other datasets (Brazdil et al. 2008). To make this recommendation, a meta-classifier is induced using meta-features describing characteristics of datasets in the meta-training set (where each meta-instance represents a dataset), and using as meta-class labels the best classification algorithm for each dataset in the meta-training set. Then, when a new dataset becomes available, the meta-classifier is used to recommend the best algorithm for that new dataset. Meta-learning approaches are useful in two main ways: 1) Since they automate the choice of the best algorithm to a new dataset, they avoid the need for running many classification algorithms on the new dataset, which is an *ad hoc* but very popular approach to choose the best classification algorithm in practice. 2) If interpretable meta-classification models are induced, they can be used to explain why a hierarchical classification algorithm recommended for a new dataset.

Meta-learning approaches can be of great use in hierarchical classification problems. In such problems it is typically difficult to choose an appropriate hierarchical classification algorithm for the task. The higher problem complexity and the usually very long run times associated with applying many hierarchical classification algorithms to a new dataset make the use of exploratory experiments more difficult.

In this work we have experimented with the following commonly used hierarchical classification algorithms: Predictive Clustering Tree (PCT) (Vens et al. 2010), Predictive Clustering Trees Ensemble (PCTEN) (Schietgat et al. 2010), and Local Hierarchical Classifiers (LHC) (Koller and Sahami 1997). We propose new meta-features for meta-learning in hierarchical classification, and also propose a new algorithm for generating classification datasets from existing ones exploiting the particularities of hierarchical classification problems. As far as we know, this is the first work to propose meta-learning for automatically recommending a hierarchical classification algorithm.

We also propose a new algorithm for generating new hierarchical classification datasets from existing ones exploiting the particularities of hierarchical classification problems. As far as we know, this is the first time meta-learning has been applied in the context of hierarchical classification.

This chapter is organised as follows: Section 7.2 presents background on meta-learning. Section 7.3 presents the definition of the meta-features used to describe the hierarchical classification datasets. Section 7.4 describes the algorithm proposed to split existing hierarchical classification datasets into new datasets (using different parts of the class hierarchy), thus greatly increasing the number of meta-instances. Section 7.5 presents the experimental setup we used to build our meta-learning framework. Section 7.6 presents the results of our meta-learning framework, including an analysis of the predictive performance of our meta-classification system and an interpretation of the meta-models generated to choose between the four above mentioned candidate hierarchical classification algorithms given the meta-features.

7.2 Background on Meta-learning

Meta-learning for algorithm recommendation, in the classification setting, is the computational task of predicting the performance of classification algorithms (representing meta-classes) given their past performance and meta-features that describe the characteristics of datasets (the meta-instances).

Broadly speaking, there are three types of meta-features: 1) dataset-derived meta-features (Peng et al. 2002), 2) landmarking meta-features (Peng et al. 2002) and 3) sampling meta-features (Leite and Brazdil 2010). The first type of meta-feature are numerical values that characterise some aspect of the dataset, such as the number of instances, the number of classes, the class distribution, and so on, which can be directly extracted from the dataset, without inducing a classification model. Landmarking meta-features are defined as features characterising some classification model induced using the dataset. Typically, this classification model must be relatively computationally inexpensive to induce and should provide insights about the classification problem at hand. The last type of meta-feature, sampling meta-features, consists of applying the classification algorithms whose performance are being predicted in a sample of the testing dataset and using the predictive performance results as meta-features.

Both approaches (landmarking and sampling) aim to extract meta-features from classification models that can be induced fast. Note, however, that landmarking approaches usually use a *fast classification algorithm* and extract meta-features from the model that was induced using the full base dataset (meta-instance). For instance, if the classification model is a decision tree, structural descriptors of the tree can be used as meta-features. On the other hand, sampling approaches can use more computationally expensive classification algorithms trained on a small sample of each base dataset (meta-instance). Usually, sampling approaches use as meta-features the predictive performance of the classification models on the dataset samples, instead of characteristics of the model. Commonly, the set of classification algorithms used in the sampling approach is the same set as the set of possible algorithm recommendations. The principle is that the performance on a dataset sample is a good predictor of the performance on the full dataset.

There are other, more refined meta-features. For instance, in (Sun and Pfahringer

2013) the authors induce a decision tree classifier at the meta-level and use the boolean value of each rule (each path from the root node to a leaf node) as a meta-feature for another meta-model. In this work the authors also propose the Approximate Ranking Trees (ART) ensemble method, which is an adaptation of the decision tree algorithms to predict algorithms' ranks. In (van Rijn et al. 2015) the authors build a learning curve by plotting predictive performance against sampling successively larger samples from the datasets (meta-instances). Next they analyse the behaviour of this learning curve to recommend the best classification algorithm for a new meta-instance.

In (Leite, Brazdil and Vanschoren 2012) authors propose an *active testing* framework to choose the best algorithm to be applied to a new meta-instance. Their algorithm works by doing several tournament-style comparisons between the current best classifier and a promising competitor. Their objective is to minimise the number of models that must be trained to get a reasonably good recommendation. Note that all these previous works addressed only the conventional classification task; unlike this work, which addresses the more challenging task of hierarchical classification.

The meta-learning approach proposed in this chapter has two goals. First, we want to build meta-classification models with a high predictive performance, in order to provide reliable algorithm recommendations to the user. Second, we want to discover general (and meaningful) relationships between the meta-features representing characteristics of the hierarchical classification datasets and the hierarchical classification algorithms (meta-classes) used in our experiments. These relationships could be used as explanations for the algorithm recommendations output by the meta-learning system. Hence, we do not employ predictive performance measures that combine runtime and predictive performance, such as the ones presented in (Abdulrahman and Brazdil 2014; Peng et al. 2002; van Rijn et al. 2015), as optimizing runtime often reduces classification performance.

Regarding related work, some works have applied automated techniques to choose the best 'flat' classification algorithm to predict the individual class nodes of hierarchical classification problems. In (Holden and Freitas 2008) the authors use a swarm intelligence algorithm to select which 'flat' classification algorithm to use, from a pool of classifiers. In (Silla Jr. and Freitas 2009, 2011b) authors apply a strategy to select both the best classification algorithm and best instance representation to improve the performance of their hierarchical classification system.

Note, however, that these approaches do not aim to select the whole hierarchical classification algorithm, instead, they try improve parts of a fixed hierarchical classification model. Also, these approaches do not build a model to encode the information of when to use a certain classifier, like we are doing. Instead they test all possible classifiers and select the one that optimizes a predictive performance measure.

7.3 Definition of the Proposed Meta-features

The proposed meta-features for describing properties of hierarchical classification datasets are divided into three broad types: simple *multi-label dataset-derived meta-features*, *hierarchical dataset-specific meta-features*, and *meta-features extracted from the landmarking PCT classification model.*

The first meta-feature type describes properties of a multi-label classification dataset (Tsoumakas, Katakis and Vlahavas 2010), without referring to hierarchical classification aspects. However, since every hierarchical classification dataset is implicitly a multi-label dataset (Silla Jr. and Freitas 2011a) (an instance is assigned class labels at multiple levels of the class hierarchy), such meta-features are still potentially useful to describe hierarchical classification datasets.

The second meta-feature type, specific for hierarchical classification datasets, describes the characteristics of the graph that represents the class hierarchy (Silla Jr. and Freitas 2011a).

The meta-features of these first two types are new meta-features, since this is the first work on meta-learning for hierarchical classification.

The third meta-feature type is generated by inducing a PCT hierarchical classification model (a decision tree) using the base dataset and extracting meta-features directly from the induced decision tree. These meta-features are not new, they are broadly based on the meta-features proposed in (Peng et al. 2002) for standard (flat) classification, but used here for the first time in the more complex scenario of hierarchical classification.

1. Multi-label dataset-derived meta-features

   (a) NumClasses – Number of class labels.

   (b) LabCard (label cardinality) – Average number of class labels per instance.

   (c) DistLabSetSize – Number of distinct label sets that occur in at least one instance.

   (d) NumFeats – Number of features.

   (e) NumInsts – Number of instances.

   (f) InstFeatRatio – Number of instances divided by the number of features.

2. Hierarchical dataset-specific meta-features

   (a) AvgDepth – The average length of all possible paths from the root to all the leaf nodes of the class hierarchy.

   (b) ClassImbal – Average class imbalance, defined as the average proportion of positive class instances across all class nodes in the hierarchy. Recall that a hierarchical classification problem can be viewed as a collection of binary classification problems, with restrictions defined by the class hierarchy.

   (c) NumLeaves – Number of leaf class labels.

   (d) HierType – Class hierarchy's structure type (tree or DAG).

   (e) AvgDegree – Average degree (number of edges) per class node.

   (f) MaxDegree – Maximum degree across all class nodes.

   (g) MaxLevelSize – Maximum number of nodes in a class level, across all levels. A class label is in the i-th level if there is a path of length "i" from the root to that class label's node. Note: for DAGs the same node may be in multiple levels.

   (h) MinLevelSize – Minimum number of class nodes in a level across all levels.

   (i) MeanLevelSize – Mean number of class nodes in a level across all levels.

   (j) LongBranch – Longest path among all possible paths from the root to a leaf class.

   (k) ShortBranch – Shortest path among all possible paths from the root to a leaf class.

   (l) MeanBranch – Mean path length among all possible paths from the root to a leaf class.

3. Meta-features extracted from the landmarking PCT hierarchical classification model

   (a) NumNodesPCT – Number of nodes in the decision tree induced by PCT.

   (b) NumLeavesPCT – Number of leaves in the decision tree induced by PCT.

   (c) MaxLevelSizePCT – Maximum number of internal nodes in a level of the decision tree induced by PCT, across all tree levels.

   (d) MeanLevelSizePCT – Mean number of class nodes in a level of the decision tree induced by PCT across all levels of the tree.

   (e) LongBranchPCT – Longest path among all possible paths from the root to a leaf node of the decision tree induced by PCT.

   (f) ShortBranchPCT – Shortest path among all possible paths from the root to a leaf node of the decision tree induced by PCT.

   (g) MeanBranchPCT – Mean path length among all possible paths from the root to a leaf node of the decision tree induced by PCT.

   (h) PercSelPCT – % of input features selected for inclusion in the decision tree induced by PCT (any feature with number of occurrences > 0 in the tree)

   (i) BalancednessPCT – This measures how distant the tree induced by PCT (tree T) is from a balanced tree, defined as:
   $balancedness(T) = \frac{(unbAvgDepth(|T|) - actualAvgDepth(T))}{(unbAvgDepth(|T|) - balAvgDepth(|T|))}$.
   Where: $unbAvgDepth(|T|)$ is the average depth of a completely unbalanced tree (worst case scenario) containing $|T|$ nodes, $balAvgDepth(|T|)$ is the average depth of the most balanced binary tree possible (best case scenario) containing $|T|$ nodes, and $actualAvgDepth(T)$ is the actual average depth of the tree T.

   This meta-feature has value 1 if T is the most balanced tree possible and value 0 if it is the most unbalanced tree possible.

In Figures 7.1 and 7.2 we illustrate graphically the characteristics of some interesting meta-features. Note that some pairs of meta-features are substantially more correlated than others, as measured by Pearson's linear correlation coefficient (r). For instance, the meta-features $NumInsts$ and $NumClasses$ are more positively correlated ($r = 0.40$), see Figure 7.1, than the pair of meta-features *InstFeatRatio* and *AvgDepth* ($r = 0.18$), see Figure 7.2. This fact will be important in Section 7.6, when we analyse the meta-features selected by the decision tree algorithm C4.5.

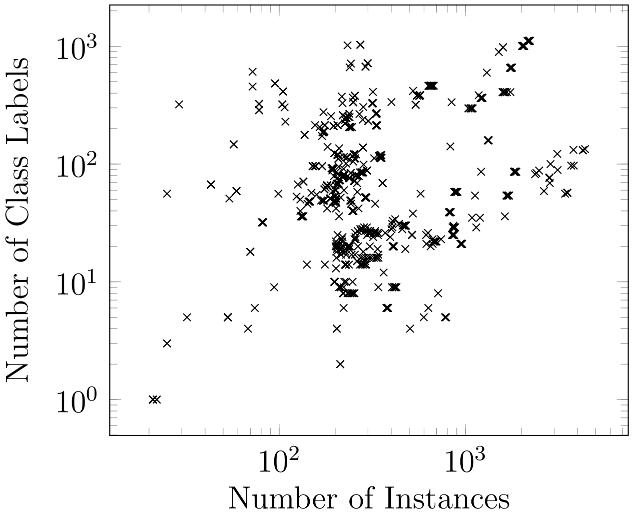


Figure 7.1: Scatter plot of the meta-features number of instances (*NumInsts*) and number of class labels (*NumClasses*), for our hierarchical classification datasets. Each ‘x’ represents a meta-instance. Note that there is a positive correlation between the two meta-features – linear correlation coefficient $r = 0.40$.

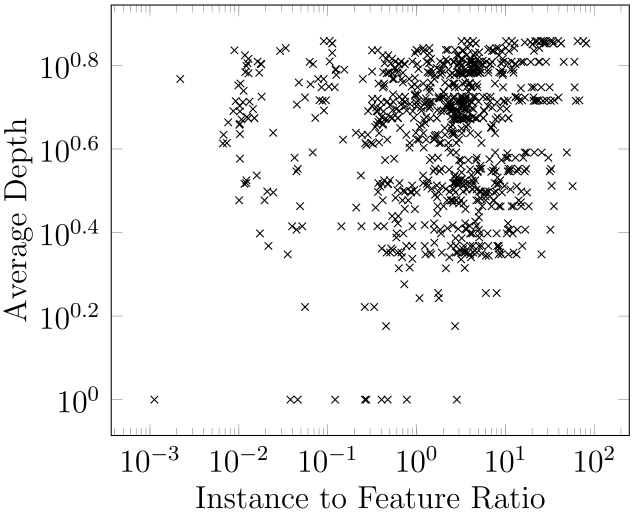


Figure 7.2: Scatter plot of the meta-features instance to feature ratio (*InstFeatRatio*) and average class depth (*AvgDepth*). Each ‘x’ represents a meta-instance. Notice that the correlation between these two meta-features ($r = 0.18$) is not so clear as the correlation in Figure 7.1.

7.4 Algorithm for Splitting the Hierarchical Datasets

One of the main challenges of applying meta-learning to data mining tasks is collecting a reasonable number of datasets to be used as meta-instances to train the meta-classifiers. This problem is exacerbated when dealing with hierarchical classification problems, as there are substantially fewer freely available datasets for this specific task than for standard ("flat") classification tasks. Actually, there is no systematic repository of hierarchical classification datasets, unlike the case for standard classification, where there are well-known dataset repositories like the UCI one (Lichman 2013).

For this reason we propose a new approach for creating a larger number of hierarchical classification datasets from an existing set of available hierarchical classification datasets. Note that the created datasets preserve part of the data contained in the original datasets, as explained below. Hence, the new datasets still contain real-world data, rather than being synthetic datasets generated in a random way. In this work we have applied this approach (formalised by Algorithm 7.1) to divide the original 42 hierarchical classification datasets into 862 new hierarchical classification datasets. Broadly speaking, for each original dataset, Algorithm 7.1 divides the existing class label hierarchy into sub-hierarchies and creates a new hierarchical classification dataset (to be used as a meta-instance) for each sub-hierarchy. Each instance in the full original dataset is present in a new hierarchical classification dataset if that instance is annotated with at least one class label from the sub-hierarchy corresponding to that new dataset.

Algorithm 7.1 requires the specification of a "spanning set" by the user. In our context, a "spanning set" is a set of class labels that Algorithm 7.1 uses to decide how to "break down" the class hierarchy: the children of the class labels in this set define the unique classes of the new sub-hierarchies, that is, these children will not be shared between the new sub-hierarchies. Note that the classes in the spanning set must be chosen according to the structure of the hierarchy: a good spanning set is one containing class labels with a reasonable number of child class labels (so that a considerable number of sub-hierarchies can be generated) and close to the hierarchy's root node (so that each generated sub-hierarchy can have a reasonable number of instances).

Algorithm 7.1 works by iterating over the children of the class labels in the "spanning set" (line 5) and checking if the current child node has less than $minDesSize$ (minimum Desirable Size) instances (line 6) – a user-defined parameter. If this is the case, the instances in the current child node are added to the temporary set *toGenerate* (line 7). Notice that if an instance belongs to multiple classes, it cannot appear multiple times in this set; this is guaranteed by the append operator (line 7). If *toGenerate* has accumulated *minDesSize* or more instances (line 8), a new hierarchical classification dataset containing the instances in *toGenerate* is created (line 9) and the algorithm proceeds to iterate over the next child node of a class node in the spanning set.

If the current child class node has *minDesSize* instances or more (i.e., the *if* test in line 6 fails), a new hierarchical classification dataset is created using that child node and its descendants (line 13). Finally, when the *for* loop ends, the algorithm checks if the current number of instances in *toGenerate* is greater than or equal to *minAccSize* (minimum Acceptable Size) – another user-defined parameter. If that is the case, a new hierarchical classification dataset is created (line 17); otherwise the instances are discarded, as we consider that a dataset with less than *minAccSize* instances does not contain enough information to be used in our experiments.

Algorithm 7.1 Split a hierarchical classification dataset into many smaller ones

```
 1: procedure SPLIT(span (The spanning set))
 2:     $minDesSize = 200$
 3:     $minAccSize = 20$
 4:     $toGenerate = \{\}$
 5:     for each $child \in span.children$ do
 6:         if $|child.instances| < minDesSize$ then
 7:             $toGenerate.append(child.instances)$
 8:             if $|toGenerate| \geq minDesSize$ then
 9:                 $generateDS(toGenerate)$
10:                 $toGenerate = \{\}$
11:             end if
12:         else
13:             $generateDS(\{child.instances\})$
14:         end if
15:     end for
16:     if $|toGenerate| \geq minAccSize$ then
17:         $generateDS(toGenerate)$
18:     end if
19: end procedure
```

In our experiments we have set the value of $MinDesSize$ to 200 and $MinAccSize$ to 20. As a result, the algorithm will first generate hierarchical datasets with at least 200 instances during the execution of the *for* loop. Then, at the end of the algorithm, it will generate at most one dataset with at least 20 instances with class labels that could not pass the $MinDesSize$ criterion, in order to avoid discarding instances unnecessarily.

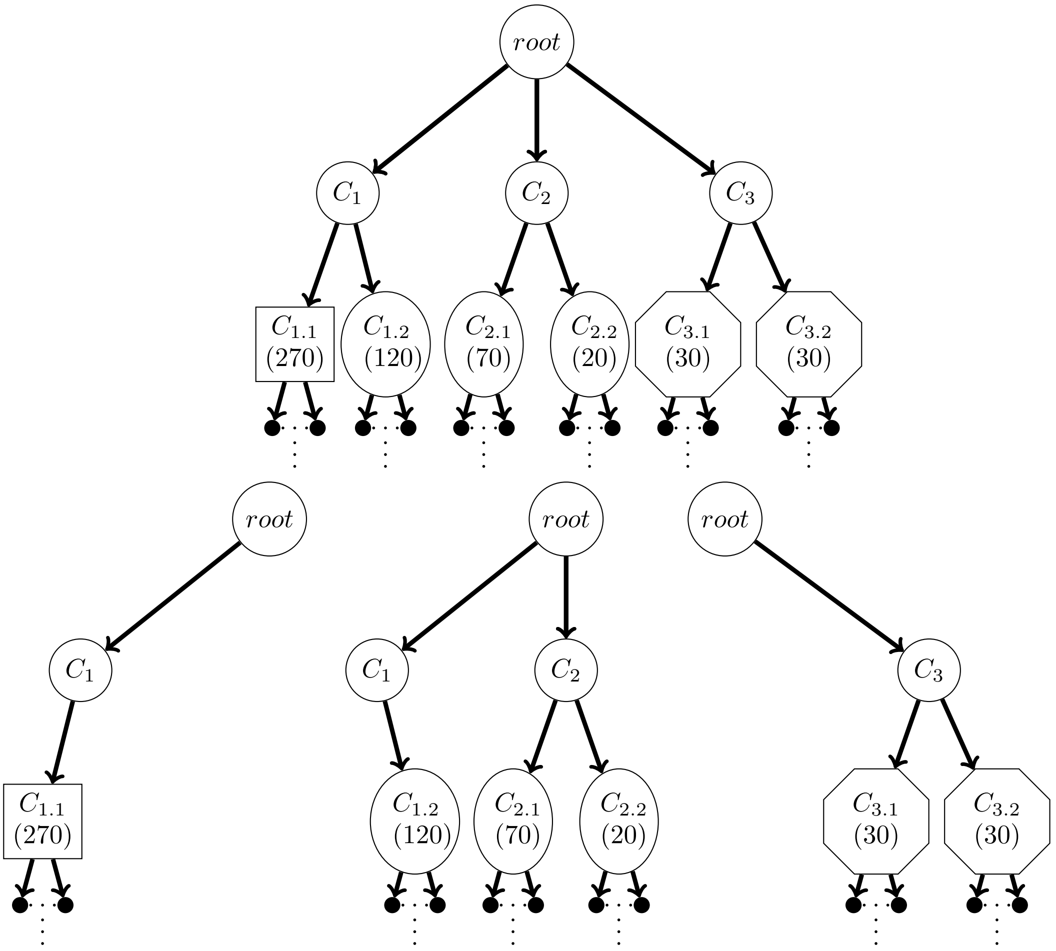


Figure 7.3: Result of applying Algorithm 7.1 on a hypothetical dataset with the hierarchy presented in the top part of the figure using a set containing class labels C_1,C_2, and C_3 as the spanning set. The bottom part of the figure shows the resulting new hierarchies. Numbers in brackets are the number of instances annotated with a particular class. Ellipses mean suppressed hierarchical classes.

Figure 7.3 shows the result of applying Algorithm 7.1 to a hypothetical class hierarchy using the set of classes $\{C_1, C_2, C_3\}$ as the spanning set. We show in brackets the number of instances belonging to a class node that is a child of some node in the spanning set. These numbers are suppressed in the other nodes because they are not relevant to Algorithm 7.1. Also, note that the leaf classes have an arbitrary number of descendants (represented by ellipses) that do not change the behaviour of the algorithm.

Line 5 of Algorithm 7.1 iterates over the children of the class labels in the spanning set. In our example, those children are classes $C_{1.1}$, $C_{1.2}$, $C_{2.1}$, $C_{2.2}$, $C_{3.1}$, and $C_{3.2}$. Starting with class $C_{1.1}$, the algorithm checks at line 6 if the class label $C_{1.1}$ annotates less than $minDesSize$ (200) instances. Because this is not the case, at line 13 the algorithm generates a new hierarchical classification dataset using the instances that are annotated with class label $C_{1.1}$. The sub class hierarchy corresponding to this dataset is presented in the leftmost graph in the bottom part of Figure 7.3.

Next, the algorithm iterates over the next class label, $C_{1.2}$. This time the class label annotates less than $MinDesSize$ instance, so line 7 is executed and the 120 instances that are annotated with the class label $C_{1.2}$ are added to the set *toGenerate*, which was previously empty. Now *toGenerate* has less than $MinDesSize$ instances, failing the *if* condition at line 8. The algorithm proceeds to iterate over the next class label, $C_{2.1}$, the *if* condition at line 6 is satisfied and the instances that are annotated with class label $C_{2.1}$ are added to the set *toGenerate*. Assuming, for simplicity, that instances do not have, at the same time, class labels $C_{1.2}$ and $C_{2.1}$, *toGenerate* now has size 190 (120 instances from $C_{1.2}$ and 70 instances from $C_{2.1}$). For this reason the *if* condition at line 8 fails again and the instances with the next class label, $C_{2.2}$, are added to the set *toGenerate*, which now has size 210. Therefore, the *if* condition at line 8 is now evaluated as true and line 9 is executed, creating a new dataset with instances annotated with classes $C_{1.2}$, $C_{2.1}$, and $C_{2.2}$ (as well as their descendants), emptying the set *toGenerate* in the next line. The sub class hierarchy associated with classes $C_{1.2}$, $C_{2.1}$, and $C_{2.2}$ is shown in the middle graph at the bottom of Figure 7.3.

Next, the algorithm iterates over the class labels $C_{3.1}$ and $C_{3.2}$. As both class labels have less than $MinDesSize$ instances, they are added to the set *toGenerate* (line 7). The list of children of the class nodes in the spanning set terminates, finishing the *for* loop (line 15). Next, the *if* condition at line 16 is evaluated to true, as the set *toGenerate* has more than $MinAccSize$ (20) instances. Therefore, line 17 is executed, generating the final dataset with 60 instances and class labels $C_{3.1}$ and $C_{3.2}$ (assuming that no instance is annotated with labels $C_{3.1}$ and $C_{3.2}$ at the same time). The subclass hierarchy containing classes $C_{3.1}$ and $C_{3.2}$ is displayed in the rightmost graph at the bottom of Figure 7.3.

Regarding related work, the idea of generating new meta-instances from existing ones, in order to substantially increase the number of meta-instances, has been explored in the very different context of flat (non-hierarchical) regression. In (Loterman and Mues 2015) the authors exchange the roles of predictive features and the continuous target variable to create more meta-instances. That is, each feature is interpreted as a target variable, using the original target and all other features as predictive features, hence creating a new dataset for each feature used as a target. However, as far as we know, our current work is the first one to propose a new algorithm to create new classification datasets from existing ones in meta-learning, using information about the hierarchical classes, as proposed in this section.

In addition, in the new datasets created in (Loterman and Mues 2015), the corresponding new regression problems are arguably very artificial, since the original features were not supposed to be used as target variables. In contrast, our proposed Algorithm 7.1 preserves the roles of class labels and features, therefore keeping the original purpose of the datasets. On the other hand, Algorithm 7.1 creates datasets that are smaller than the original datasets; in contrast with (Loterman and Mues 2015), which preserves the size of the original datasets.

7.5 Experimental Setup

Our meta-learning approach consists of inducing a multi-class meta-classifier using the meta-features presented in Section 7.3 as predictive attributes and the name of the best hierarchical classifier for a particular meta-instance (hierarchical classification dataset) as the meta-classes to be predicted. We use 10-fold cross validation to estimate the performance of hierarchical classification algorithms. This multi-class meta-classifier must be capable of outputting a score that represents the likelihood of a meta-instance belonging to each one of the meta-classes (a hierarchical classification algorithm). The scores are used to determine the recommended algorithm ranking, i.e., for each meta-instance, the hierarchical classification algorithm with the highest score is ranked first, the hierarchical classification the with the second highest score is ranked second, and so on.

In order to induce the meta-classifier, we have selected two multi-class classification algorithms for our experiments: the Support Vector Machine (SVM) algorithm (Boser, Guyon and Vapnik 1992) and the C4.5 decision tree algorithm (Quinlan 1993). These algorithms have complementary characteristics: the SVM algorithm usually produces models having high predictive accuracy, but difficult to interpret; while the C4.5 algorithm often produces models that are usually associated with good interpretability but with inferior predictive accuracy when compared with SVM – although of course the issue of which algorithm is more accurate depends on the underlying dataset. Actually, in our experiments J48 performed better than SVM, as reported later. We have used the J48 implementation of C4.5 from the Weka data-mining framework (Hall et al. 2009) and the SVM from LibSVM (Chang and Lin 2011). We have used the default parameters for the J48 algorithm and the *Gaussian* kernel for the SVM algorithm, using an internal cross validation procedure to select the value of parameter σ, varying its value from 0.0 to 1.0, with 0.1 increments.

In summary, the SVM algorithm implicitly maps the original problem to a high-dimensional space using kernel functions and finds a meta-class-separation hyperplane on this space that minimises classification error. The J48 algorithm builds a decision tree that recursively divides the data using a feature-based condition. Several conditions are tested and the algorithm selects the one that better separates the meta-classes of the meta-instances. We call our meta-learner using the J48 algorithm the Decision Tree Meta-Ranker (DTMR) and our meta-learner using the SVM algorithm the SVM Meta-Ranker (SVMMR).

We have used the three predictive performance measures defined in Section 6.2 to define our meta-classes. Because the measures have different biases, each one leads to a different ranking for the hierarchical classifiers. Therefore we created three meta-datasets, one for each predictive performance measure. That is, the meta-features present in the three meta-datasets are the same, but the meta-classes are different: The meta-class of each meta-instance in each of the three meta-datasets is the hierarchical classifier with the best predictive performance for the particular measure we are considering ($AU(\overline{PRC})$, $\overline{AUPRC_w}$, and $\overline{AUPRC}$). Recall that the 4 hierarchical classifiers used as meta-classes are PCT, PCTEN, LHC and HDN-nHPC (see Section 6.2).

In addition to the DTMR and SVMMR rankers, we also test two simple baselines: the first is a simple naive classification algorithm called the Prior Ranker (PR). This algorithm outputs as the predicted rank the classifier rank observed in the training set, that is, the first ranked hierarchical classifier is the one with more wins in the training set, the classifier ranked second is the next classifier in terms of wins, and so on. The second baseline is the Random Ranker (RR), which randomly assigns the rankings of the hierarchical classifiers to each meta-instance.

7.6 Experimental Results

In this section we present the meta-learning results of applying the approach for inducing and combining our meta-classifiers (as described in the previous section) to our meta-instances generated using Algorithm 7.1 (described in Section 7.4). Subsection 7.6.1 presents the predictive performance results, comparing our method with the two aforementioned simple baselines using 10 fold cross-validation and two measures of predictive performance: the *Spearman's rank correlation coefficient* (Gautheir 2001), which is often used in meta-learning research, and the more traditional *accuracy* performance measure. In Subsection 7.6.2 we interpret the meta-models induced by J48 using the whole meta-dataset to try to get useful knowledge about which characteristics of hierarchical classification datasets are good predictors of the best hierarchical classification algorithm for a new dataset.

7.6.1 Meta-Learning Performance Evaluation

We use two measures of predictive performance. The first one is the traditional *accuracy* measure, defined as the number of correct predictions divided by the total number of predictions. We consider that a prediction is correct if the predicted best hierarchical classifier (the meta-class) is the actual best classifier, for a given dataset (meta-instance). If there is a tie in the meta-classifier's ranks for the predicted best classifiers, we use the tied meta-class with the highest prior probability. If there is a tie in the actual best hierarchical classifiers, predicting any of the tied hierarchical classifiers as the best classifier is considered correct.

The second measure of predictive performance we use is the mean *Spearman's rank correlation coefficient* ($\bar{R}$) across all the datasets. $\bar{R}$ measures the agreement between the ranking of the base hierarchical classification algorithms predicted by the meta-classifier and their corresponding actual ranking, for each dataset, and it is defined as (Gautheir 2001):

$$\bar{R} = \frac{1}{J}\sum_{j=1}^{J}\left[1 - \frac{6\sum_{a=1}^{A}(pr_{j,a} - ar_{j,a})^2}{A^3 - A}\right]. \tag{7.1}$$

Where J is the number of meta-instances (hierarchical classification datasets), A is the number of base hierarchical classification algorithms, $pr_{j,a}$ is the predicted rank for the a-th base hierarchical classification algorithm in the j-th meta-instance, and $ar_{j,a}$ is the actual rank for the a-th base hierarchical classification algorithm in the j-th meta-instance. The multi-class meta-classification models we are using do not output meta-class ranks, however they output scores, which can be simply transformed to ranks by ordering the scores from the largest (most probable meta-class) to the smallest (least probable meta-class) one.

The $\bar{R}$ correlation measure lies in the interval $[-1, 1]$, where $\bar{R} = 1$ means that there is a perfect agreement between the predicted and actual ranks, $\bar{R} = 0$ means that there is no correlation between the predicted and actual ranks, and $\bar{R} = -1$ means that there is a perfect disagreement between the predicted and actual ranks. This ranking correlation coefficient has been used in several works dealing with ranking-based meta-learner evaluation in the classification task, e.g.: (Peng et al. 2002; Sun and Pfahringer 2013; Brazdil, Soares and Da Costa 2003).

Table 7.1 shows the results of applying the two more sophisticated meta-rankers (DTMR and SVMMR), the Prior Ranker (PR), and the Random Ranker (RR) to our three meta-datasets. We show the predictive performance results using two different measures, the previously defined R rank correlation coefficient and the more common *accuracy measure*. Recall that a 'perfect' ranker (a ranker that always predict the ranking of the classification algorithm correctly) would have a R rank correlation and accuracy of 1.0.

Table 7.1: Rank correlation coefficient and accuracy results of applying the DTMR, SVMMR, PR, and RR meta-rankers to our meta-datasets, one meta-dataset for each hierarchical classification version of the AUPRC measure, using the 10-fold cross-validation procedure.

	R rank correlation				Accuracy			
	DTMR	SVMMR	PR	RR	DTMR	SVMMR	PR	RR
$AU(\overline{PRC})$	0.5197	0.3628	0.3691	0.0325	0.5821	0.4648	0.3745	0.2498
$\overline{AUPRC_w}$	0.3915	0.2013	0.2234	0.0147	0.5130	0.3933	0.3382	0.2722
$\overline{AUPRC}$	0.4298	0.2240	0.2066	0.0218	0.5306	0.4342	0.3977	0.2827

Table 7.1 clearly shows that the DTMR meta-ranker is superior to every other meta-ranker we tested, including the SVMMR meta-ranker, surprisingly. We have applied pairwise *paired t-tests* to the results of the 10 folds of the cross-validation procedure, as suggested in (Japkowicz and Shah 2011), to compare the results of DTMR against each of the other three meta-rankers. Using paired t-tests is recommended when comparing one baseline learning algorithm (in this case the DTMR) to a set of other learning algorithms using one dataset when using reasonably-sized datasets. The test rejected the null hypothesis of classifier equivalence in every

comparison, with $\alpha = 0.05$, with all p-values < 0.0005. We attribute the poor performance of SVMMR to the sensitivity of kernel choice. Note that we have not tried to optimise the type of kernel in this work, we leave this task as future work; but we have optimised the choice of the σ parameter for the Gaussian Kernel using internal cross-validation, as mentioned earlier. In contrast, we have not optimised any J48 parameter – its default parameters were robust in our experiments.

7.6.2 Interpreting the Meta-Classification Models

The meta-models induced to predict which hierarchical classification algorithm (meta-class) is more accurate in each dataset (meta-instance) using the J48 decision tree algorithm (Witten and Frank 2000) are shown in Figures 7.4, 7.5, and 7.6, for the three predictive performance measures we have used, respectively: $AU(\overline{PRC})$, $\overline{AUPRC_w}$ and $\overline{AUPRC}$. These meta-models were induced by J48 using the whole meta-dataset, maximising J48's potential to find interesting meta-classification rules. We chose to interpret the J48 model instead of the SVM model because the J48 model is much easier to interpret and it has achieved much better predictive performance in comparison with the SVM model. In addition to showing each meta-model, we select some interesting meta-rules, interpret their meaning and, when possible, compare them with similar meta-rules found when using different predictive performance measures, thus reaching conclusions across the three measures.

In Figures 7.4, 7.5, and 7.6, each line of the presented meta-models represents a decision split, that is, a condition that must be satisfied by the meta-feature of a meta-instance in order for it to be passed to the next decision split, eventually reaching a leaf node, when a meta-classification is made. On the leaf nodes of the meta-model we display in general two values: the first is the total number of meta-instances classified by the leaf node, the second is the number of meta-instances misclassified by that leaf node. If this second value is not shown, it means there is no misclassified meta-instance in that node.

In the next section we discuss the quality of some meta-classification rules in terms of their *precision* and *recall*. Precision is the number of correct predictions (*true positive* predictions) made by the rule divided by the number of meta-instances covered by the rule. Recall is the number of correct predictions made by the rule divided by the total number of meta-instances with the meta-class predicted by the rule.

Interpreting the meta-model for the $AU(\overline{PRC})$ measure

```
InstFeatRatio ≤ 0.02: PCT (43.0/4.0)
InstFeatRatio > 0.02
|   NumFeats ≤ 551
|   |   HierType = Tree
|   |   |   AvgDepth ≤ 2.19: PCTEN (29.0/9.0)
|   |   |   AvgDepth > 2.19: LHC (228.0/113.0)
|   |   HierType = DAG
|   |   |   AvgDegree ≤ 2.58
|   |   |   |   AvgDegree ≤ 2.36: PCTEN (21.0/5.0)
|   |   |   |   AvgDegree > 2.36
|   |   |   |   |   NumNodesPCT ≤ 2: PCT (11.0/1.0)
|   |   |   |   |   NumNodesPCT > 2: HDN-nHPC (15.0/7.0)
|   |   |   AvgDegree > 2.58
|   |   |   |   NumInsts ≤ 1077: PCTEN (390.0/173.0)
|   |   |   |   NumInsts > 1077: LHC (72.0/4.0)
|   NumFeats > 551: PCTEN (53.0/8.0)
```

Figure 7.4: Meta-model generated to classify the meta-instances into meta-classes PCTEN, PCT, LHC, and HDN-nHPC, when considering the $AU(\overline{PRC})$ measure.

Figure 7.4 shows the meta-model induced when considering the $AU(\overline{PRC})$ measure. The first line of the model encodes the meta-rule: “If the instance to feature ratio ($InstFeatRatio$) is smaller than or equal to 0.02, that is, the base dataset has on the order of one instance for every 50 or more features, use the PCT classifier”. This meta-rule has high precision (0.91), much higher than the *a priori* probability for the ‘PCT’ meta-class (0.12). The recall of this meta-rule is also reasonably good, capturing 38% (39 out of 103) of all meta-instances annotated with the ‘PCT’ meta-class label. This meta-rule indicates that the decision tree-based PCT algorithm deals well with datasets with relatively few instances and many features, which seems due to its implicit class hierarchy-aware feature selection procedure. That is, by finding successive conditions that divide the set of meta-instances based on their different hierarchical classes well, the decision tree performs feature selection by analysing each feature’s predictive power across a large set of hierarchical classes, instead of analysing just one class at a time, like the LHC and HDN-nHPC algorithms do.

Although the advantage of the PCT classifier over the LHC and HDN-nHPC classifiers is clear when the instance to feature ratio is so small, it is not so clear why the PCTEN classifier did not perform so well on the datasets where

InstFeatRatio is smaller than or equal to 0.02. Upon further analysis, we have concluded that such a low *InstFeatRatio* is also correlated with a simpler classification problem, that is, hierarchical classification datasets (meta-instances) with low *InstFeatRatio* also tend to have much lower average values for *NumClasses*, *NumInst*, *NumLeaves*, and *DistLabSetSize*. This justifies why PCTEN was not the best performing algorithm for these datasets, i.e., given the relative simplicity of these datasets, the power of the PCTEN ensemble is not needed to maximise predictive performance. To show this point, Table 7.2 shows the average values of the previously mentioned meta-features considering the full meta-dataset and the subset of meta-instances with *InstFeatRatio* ≤ 0.02.

Table 7.2: Mean value of some meta-features (first column) in the whole meta-dataset (second column) and in the leaf node in the first line of the meta-model shown in Figure 7.4, where all meta-instances have *InstFeatRatio* ≤ 0.02 (third column). The last column shows the 95% confidence interval, assuming a normal distribution, of each meta-feature in the set of meta-instances of that leaf node.

Feature Name	Mean in meta-dataset	Mean in leaf	95% conf. inter. in leaf
InstFeatRatio	5.63	0.011	$[0.010, 0.012]$
NumClasses	152.27	85.95	$[64.8, 107.1]$
NumInst	530.52	217.57	$[196.2, 239.0]$
NumLeaves	47.08	23.95	$[19.1, 28.8]$
DistLabSetSize	72.52	31.24	$[24.9, 37.6]$

The last line of the meta-decision tree shown in Figure 7.4 contains another interesting meta-rule: ‘if the instance to feature ratio (*InstFeatRatio*) is greater than 0.02, and the number of features (*NumFeats*) is greater than 551, use PCTEN. This points to the advantage of using the PCTEN algorithm when the problem is more complex (higher number of features), but with enough instances to learn from (with an instance to feature ratio that is not too small). This meta-rule also has a high precision of 0.85, much higher than the *a priori* meta-class probability of 0.40. It also has a reasonable recall of 13% of all meta-instances annotated with the ‘PCTEN’ meta-class label. The next meta-rule from Figure 7.4 that we would like to highlight is as follows.

```
IF (InstFeatRatio > 0.02) AND (NumFeats ≤ 551) AND (HierType = DAG)
   AND (AvgDegree > 2.58) AND (NumInst > 1077) THEN LHC (72.0/4.0)
```

Overall, this meta-rule seems to suggest that the LHC classifier is clearly recommended when the problem is moderately difficult (relatively many instances, not very many features, a DAG class hierarchy and a class hierarchy with large average degree). Note that this meta-rule has a precision of 94% on the meta-dataset and a reasonable recall (68 meta-instances), capturing 22% of all meta-instances annotated with the 'LHC' meta-class label.

Lastly, we highlight the following meta-rule for the HDN-nHPC meta-class.

```
IF (InstFeatRatio > 0.02) AND (NumFeats ≤ 551) AND (HierType = DAG)
  AND (AvgDegree ≤ 2.58) AND (AvgDegree > 2.36)
  AND (NumNodesPCT > 2) THEN HDN-nHPC (15.0/7.0)
```

This meta-rule captures meta-instances with $InstFeatRatio$ greater than 0.02, like the PCTEN. Note, however that the number of features is limited to 551 (NumFeats $\leq$ 551), and the average degree of the nodes in the class hierarchy is in the interval $(2.36, 2.58]$, which is close to the mean of the AvgDegree meta-feature (2.69) on the entire meta-dataset. In addition, this meta-rule captures meta-instances whose landmarking PCT is not too small (NumNodesPCT $>$ 2). Note that the recall of this meta-rule is not high (7%), but its precision is reasonably high (0.53), compared to the *a priori* meta-class probability of 0.12. In summary this meta-rule seems to capture meta-instances with considerable complexity with non-trivial landmarking PCT models. Note that this is the only meta-rule that has predicted the HDN-nHPC meta-label, and has a relatively poor quality in terms of precision and recall when compared to the other meta-rules, which indicates the difficulty of finding good meta-rules for predicting this particular meta-class label. Even thought the quality of this rule is arguable, we chose to keep for the sake of completeness.

Interpreting the meta-model for the $\overline{AUPRC_w}$ Measure

```
InstFeatRatio ≤ 0.02: PCT (43.0/2.0)
InstFeatRatio > 0.02
|   MeanLevelSizePCT ≤ 2.5
|   |   ClassImbal ≤ 0.60
|   |   |   NumInsts ≤ 43: LHC (10.0)
|   |   |   NumInsts > 43
|   |   |   |   ClassImbal ≤ 0.06: HDN-nHPC (15.0/4.0)
|   |   |   |   ClassImbal > 0.06
|   |   |   |   |   AvgDegree ≤ 3.69: PCT (336.0/183.0)
|   |   |   |   |   AvgDegree > 3.69: LHC (13.0/3.0)
|   |   ClassImbal > 0.60: PCTEN (16.0/2.0)
|   MeanLevelSizePCT > 2.5
|   |   LongBranchPCT ≤ 21
|   |   |   MinLevelSize ≤ 1
|   |   |   |   NumFeats ≤ 2425: LHC (382.0/190.0)
|   |   |   |   NumFeats > 2425: PCTEN (17.0/3.0)
|   |   |   MinLevelSize > 1: PCT (16.0/7.0)
|   |   LongBranchPCT > 21: HDN-nHPC (14.0)
```

Figure 7.5: Meta-model generated to classify the meta-instances into meta-classes PCTEN, PCT, LHC, and HDN-nHPC, when considering the $\overline{AUPRC_w}$ measure.

Analysing the meta-model shown in Figure 7.5 we observe that, interestingly, the same condition that the J48 algorithm selected to predict the 'PCT' meta-class using the $AU(\overline{PRC})$ measure was selected again to predict the 'PCT' meta-class for the $\overline{AUPRC_w}$ measure. It is interesting that both models chose this condition as the root of the meta-decision tree, highlighting the importance of the meta-feature $InstFeatRatio$. In addition, note that the same threshold of 0.02 was consistently chosen for the $InstFeatRatio$ meta-feature in both Figure 7.4 and Figure 7.5. This meta-rule has a precision of 0.95, much higher than the *a priori* meta-class probability of 0.28. In addition, this meta-rule has a similar number of 41 true positive instances as the meta-rule for the $AU(\overline{PRC})$ measure. This coverage, however, represents a recall of only 17% of the meta-instances annotated with the 'PCT' meta-class label, rather than 38% as in the corresponding meta-rule for the $AU(\overline{PRC})$ measure. This is because PCT performed better when considering the $\overline{AUPRC_w}$ measure, being ranked first in more base datasets.

For predicting the PCTEN meta-class, we highlight the following interesting meta-rule in Figure 7.5:

```
IF (InstFeatRatio > 0.02) AND (MeanLevelSizePCT  ≤ 2.5)
   AND (ClassImbal > 0.60) THEN PCTEN (16.0/2.0)
```

This meta-rule, although different from the meta-rule predicting PCTEN highlighted for the $AU(\overline{PRC})$, captures a broadly similar type of classification problem: like the meta-rule predicting PCTEN for the $AU(\overline{PRC})$ measure, the condition "InstFeatRatio > 0.02" was selected as the root of the PCT meta-model, meaning that the PCTEN algorithm is recommended when the ratio of the number of instances divided by the number of features is not very small. The other two conditions point out to a complex hierarchical classification problem: the condition "ClassImbal > 0.60" captures classification problems with a relatively large degree of class imbalance and the condition "MeanLevelSizePCT ≤ 2.5" captures problems where the PCT landmark tends to be small, indicating that the model is underfit for the base hierarchical classification dataset. Note that the condition "MeanLevelSizePCT ≤ 2.5" might also capture problems where the PCT landmark is unbalanced, having few nodes per level but being, possibly, very deep. This, however, was not the case, as the average longest branch size (LongBranchPCT) of the PCT landmark models when "MeanLevelSizePCT ≤ 2.5" is just 2.0, much smaller than the longest branch size of the PCT landmark models when "MeanLevelSizePCT > 2.5" (8.6). This meta-rule has a high precision (0.86) compared to the *a priori* probability of the 'PCTEN' meta-label (0.28), but low recall, capturing only 8% of all meta-instances annotated with the 'PCTEN' meta-class label.

Next, we analyse an interesting meta-rule for recommending the LHC algorithm in Figure 7.5:

```
IF (InstFeatRatio > 0.02) AND (MeanLevelSizePCT > 2.5)
  AND (LongBranchPCT ≤ 21) AND (MinLevelSize ≤ 1)
  AND (NumFeats ≤ 2425) THEN LHC (382.0/190.0)
```

The first condition of this meta-rule is the same as in the meta-rule predicting the PCTEN meta-class label. The second condition (MeanLevelSizePCT > 2.5) uses the same meta-feature, but the complementary range of its values, by comparison with the meta-rule predicting PCTEN – which involves the condition MeanLevelSizePCT ≤ 2.5. Hence, the meta-rule predicting LHC refers to landmark PCT models with larger numbers of nodes across the levels of the PCT tree, suggesting the base dataset is more complex. Interestingly, the next condition

(LongBranchPCT $\leq$ 21) captures meta-instances whose PCT landmark models have the longest branch smaller than or equal to 21 nodes. Unfortunately, the condition (MinLevelSize $\leq$ 1) does not inform us if the classification problem is complex or simple, it merely inform us that at least one level of the class hierarchy has one class node. Notice that the 'less than' part of this condition is superfluous, since there are no meta-instances with less then 1 node in a given level of the class hierarchy.

The last condition points out that when there are not too many features (2425 or less), it is recommended using LHC instead of PCTEN. This is consistent with the meta-rule predicting LHC for the $AU(\overline{PRC})$ measure, which also recommends using LHC when the number of features is smaller than 551, $InstFeatRatio > 0.02$ (as in the current meta-rule), and other conditions are satisfied. This meta-rule has a very high recall, capturing 79% of the meta-instances annotated with the meta-class label 'LHC', and has a reasonable precision: about 0.50, substantially higher than the *a priori* probability for the meta-class label LHC, which is 0.32.

The last meta-rule we would like to present is the one predicting the HDN-nHPC meta-label:

```
IF (InstFeatRatio > 0.02) AND (MeanLevelSizePCT > 2.5)
  AND (LongBranchPCT > 21) THEN HDN-nHPC (14.0)
```

The first condition of this meta-rule is the same as the one for predicting the PCTEN meta-class label. Broadly speaking, the second condition captures meta-instances with large PCT landmark models, indicating that they may be over-fitting the data. This second condition uses the same meta-feature used in the meta-rules predicting LHC and PCTEN. The difference is that the meta-rule predicting PCTEN uses the "$\leq$ 2.5" range of values, whist the meta-rules predicting LHC and HDN-nHPC use the "> 2.5" range of values. The last condition checks if the landmark PCT model is deep (its longest branch has more than 21 conditions), indicating that HDN-nHPC is more suited when the landmark PCT model is large and the problem is complex. The precision of this rule is 1.0 (perfect predictive accuracy). Its recall, on the other hand, is relatively low (8%).

Interpreting the meta-model for the $\overline{AUPRC}$ Measure

```
NumInsts ≤ 667
|   PercSelPCT ≤ 0.0004
|   |   ClassImbal ≤ 0.60: PCT (186.0/39.0)
|   |   ClassImbal > 0.60: PCTEN (16.0/3.0)
|   PercSelPCT > 0.0004
|   |   meanBranchPCT ≤ 2.4
|   |   |   AvgDegree ≤ 3.5: PCT (202.0/101.0)
|   |   |   AvgDegree > 3.5: PCTEN (25.0/9.0)
|   |   meanBranchPCT > 2.4
|   |   |   LabCard ≤ 2.7: PCT (20.0/4.0)
|   |   |   LabCard > 2.7: PCTEN (233.0/108.0)
NumInsts > 667
|   ShortBranch ≤ 4
|   |   InstFeatRatio ≤ 21.2
|   |   |   PercSelPCT ≤ 0.003: PCTEN (12.0/3.0)
|   |   |   PercSelPCT > 0.003
|   |   |   |   AvgDepth ≤ 7.12
|   |   |   |   |   LongBranchPCT ≤ 10: LHC (74.0/37.0)
|   |   |   |   |   LongBranchPCT > 10: HDN-nHPC (27.0/9.0)
|   |   |   |   AvgDepth > 7.12: LHC (10.0)
|   |   InstFeatRatio > 21.17
|   |   |   ShortBranchPCT ≤ 2: PCT (10.0/1.0)
|   |   |   ShortBranchPCT > 2: LHC (29.0/10.0)
|   ShortBranch > 4: LHC (18.0)
```

Figure 7.6: Meta-model generated to classify the meta-instances into meta-classes PCTEN, PCT, LHC, and HDN-nHPC, when considering the $\overline{AUPRC}$ measure.

Analysing the meta-classification model for the $\overline{AUPRC}$ measure as shown in Figure 7.6, we can draw broadly similar conclusions to the conclusions derived for measures $AU(\overline{PRC})$ and $\overline{AUPRC}_w$.

Namely, when the problem has characteristics that are commonly recognised to harm predictive performance (e.g. the class distribution is very imbalanced, there are relatively few instances), PCTEN is more suited to solve the problem than PCT. The following two meta-rules show this behaviour. These two meta-rules are represented in decision tree format in order to highlight the fact that they differ only in the value range of the meta-feature ClassImbal (in the third condition) and in the predicted meta-class label.

```
NumInsts ≤ 667
|    PercSelPCT ≤ 0.0004
|    |    ClassImbal ≤ 0.60: PCT (186.0/39.0)
|    |    ClassImbal > 0.60: PCTEN (16.0/3.0)
```

Note that these two meta-rules have a condition covering datasets with a relatively small number of instances (≤ 667) and a condition covering datasets where a very small percentage of features is selected by the PCT landmark model (PercSelPCT ≤ 0.0004).

The meta-rule leading to the prediction of the 'PCTEN' meta-class label has the precision of 0.81, much higher than the *a priori* 'PCTEN' meta-class label probability of 0.20; but a low recall, only capturing 8% of the meta-instances annotated with the 'PCTEN' meta-class label. However, we consider that the precision is high enough for this meta-rule to be explicitly mentioned.

Conversely, the meta-rule leading to the prediction of the 'PCT' meta-class label has opposite characteristics: a relatively high recall (capturing 44% of all meta-instances annotated with the meta-class label 'PCT') and a slightly lower precision (0.79), although that precision is still relatively high, compared to the *a priori* 'PCT' meta-class label probability (0.39).

We present the following meta-rule to predict the LHC meta-label.

```
IF (NumInsts > 667) AND (ShortBranch > 4) THEN LHC (18.0)
```

Once again, the meta-rule predicts the LHC meta-class label when the problem tends to be simpler. In this case, a problem with more instances and fewer class labels, since meta-instances with the shortest branch having more than 4 class-labels correlate strongly with meta-instances with fewer classes. The average number of classes (NumClasses meta-features) in the meta-instances when ShortBranch $>$ 4 is 59, much smaller than the average number of classes when ShortBranch $\leq$ 4, which is 156.5. This apparently counter-intuitive result is due to the fact that the Gene Ontology datasets have more leaf hierarchical classes at shallower levels, and also have, overall, more hierarchical classes. This meta-rule has the perfect precision of 1.0, and a reasonable recall of 10%.

Finally, we highlight the following meta-rule that predicts the HDN-nHPC meta-class label.

```
IF (NumInsts > 667) AND (ShortBranch ≤ 4) AND (InstFeatRatio ≤ 21.2)
  AND (PercSelPCT > 0.003) AND (AvgDepth ≤ 7.12)
  AND (LongBranchPCT > 10) THEN HDN-nHPC (27.0/9.0)
```

This long meta-rule is difficult to interpret, since it comprises conditions that cover both complex and simple hierarchical classification problems. For instance, the condition “NumInsts $>$ 667” and “AvgDepth $\leq$ 7.12” cover simpler meta-instances (base datasets). The condition “InstFeatRatio $\leq$ 21.2” seem to cover more complex classification problems. However, upon further analysis, this condition covers the vast majority (94%) of meta-instances, so this condition actually removes the 6% of the easiest meta-instances according to the InstFeatRatio meta-feature. The condition “ShortBranch $\leq$ 4” covers more complex meta-instances, as previously explained in the interpretation of the meta-rule for the LHC algorithm. The other two conditions (PercSelPCT $>$ 0.003) and (LongBranchPCT $>$ 10), once again, clearly cover meta-instances with complex PCT landmark models. Note that the condition (PercSelPCT $>$ 0.003) is strongly related with the PCT landmark model size; meta-instances that satisfy this condition have an average NumNodesPCT of 33.2, much larger than the average NumNodesPCT of 5.8 for the meta-instances satisfying the complement of this condition (PercSelPCT $\leq$ 0.003).

Note that, once again, the recall of this meta-rule is not high (19%). Its precision is relatively high (0.66), compared to the *a priori* HDN-nHPC meta-class label probability of 0.11. Also, again, this is the only meta-rule that has predicted the HDN-nHPC meta-label for the $\overline{AUPRC}$ measure, which indicates the difficulty of finding good rules for predicting this particular meta-class label.

7.7 Conclusions

In this chapter we have proposed the first meta-learning approach for hierarchical classification and proposed new meta-features, specific for the hierarchical classification task. We have also proposed a new algorithm to generate new hierarchical classification datasets from existing ones. This algorithm tackles the main issue of applying meta-learning in hierarchical classification problems, the limited number of datasets, by using the class hierarchy to generate new datasets.

Summarising our interpretation findings, we have concluded that overall, at a high level of abstraction, PCTEN (an ensemble algorithm) is more suited to solve the target hierarchical classification problem when the corresponding dataset is relatively complex. Broadly speaking, the PCT algorithm seems to perform better when the problem is simpler and has more features, while the LHC algorithm performs better when the number of features is smaller. The relative superiority of the HDN-nHPC algorithm was the most difficult to predict, but it appears that the HDN-nHPC algorithm may be indicated when the landmark PCT model for the base dataset is relatively large.

Chapter 8

Conclusions and Future Research

In this chapter we present a summary of the contributions of this thesis in Section 8.1 and possible lines of future research in Section 8.2.

8.1 Summary of Contributions

This interdisciplinary work has proposed several contributions to the area of hierarchical classification and applications in bioinformatics, specifically in the classification of ageing-related genes/proteins. In essence, these contributions are as follows:

1. We have proposed a modification of the Extended Local Hierarchical Naive Bayes (ELHNB) algorithm, greatly improving its running time without sacrificing predictive performance. This work is presented in Chapter 4.

2. We have created new hierarchical classification datasets for ageing research, including the development of a new feature type, the KEGG Influence (KEGGI). The creation of the datasets and the new feature type are described in sections 6.1.2 and 6.1.3.

3. We have created four new algorithms for hierarchical classification, as described in Chapter 5. These algorithms are evaluated in Chapter 6.

4. We have also proposed the first meta-learning approach for automatically recommending the best hierarchical classification algorithm to a new hierarchical classification dataset. We report the results of this study in Chapter 7.

5. We have performed a biological interpretation of some hierarchical classification models generated to classify ageing-related proteins. This interpretation has been done in collaboration with a biologist expert on ageing, Dr. Jennifer M. A. Tullet (University of Kent), as presented in Section 6.4. We have also performed a literature review, in collaboration with the ageing biology expert Dr. João Pedro Magalhães (University of Liverpool), on the current state of the interpretation of supervised machine learning algorithms' results on ageing research, as described in Section 3.3.

Next, we describe each one of these contributions in more detail.

8.1.1 Reducing the Runtime of the ELHNB Algorithm

We have substantially reduced both the training and testing run times of the ELHNB (Extended Local Hierarchical Naive Bayes) algorithm. This algorithm is an extension of the Naive Bayes algorithm for hierarchical classification that considers the state of the neighbourhood of a class variable in the class hierarchy to classify an instance. This algorithm can be seen as a collection of local Naive Bayes classifiers that take into account the prediction of other classifiers in the class hierarchy.

We have exploited the fact that this algorithm was designed to work with classification problems where each instance is associated with a single path from the root node to a leaf node in the class hierarchy – called single path, mandatory leaf class prediction problems – to substantially reduce the runtime of both its training and testing phases. The runtime of the modified algorithm was significantly better than the standard version, with equivalent predictive performance.

8.1.2 Creation of Hierarchical Classification Datasets of Ageing-Related Genes

We have created 20 new ageing-related hierarchical classification datasets to test our new hierarchical classification algorithms. These new datasets were described in sections 6.1.2 and 6.1.3 and are freely available at `https://dx.doi.org/10.13140/RG.2.2.34027.23843`.

We have also proposed a new type of feature relevant for gene/protein function prediction. This type of feature is derived from KEGG biological pathways. This feature measures the influence that a certain protein (an instance) has in a given

KEGG pathway as a whole – producing one such feature for each of the KEGG pathways being considered. This feature type has good predictive power and can be used in problems where it is important to have an interpretable classification model. This new feature type was described in Section 6.1.3.

8.1.3 Creation of new Algorithms for Hierarchical Classification

We have developed four new algorithms for hierarchical classification: the Hierarchical Dependence Network (HDN) algorithm, the HDN based on finding non-Hierarchically related Predictive Classes (HDN-nHPC), and two hybrids involving the well-known Predictive Clustering Tree (PCT) algorithm, namely PCT combined with HDN (HDN-PCT) and PCT combined with Local Hierarchical Classifiers (PCT-LHC).

We have concluded that one of our four new algorithms (the HDN-nHPC algorithm) outperforms all other seven algorithms in terms of average rank across 42 hierarchical classification datasets (according to one out of the three hierarchical predictive performance measures). The other algorithms compared against HDN-nHPC included the state-of-the-art hierarchical classification algorithm PCTEN (an Ensemble of PCT classifiers). In terms of *statistically significant* ranking differences, our HDN-nHPC algorithm outperforms four other hierarchical classification algorithms, including the popular PCT algorithm, in one out of the three hierarchical predictive performance measures. The HDN-nHPC algorithm uses Dependence Networks (DN), an under-explored type of probabilistic graphical model, to model class dependencies that are not specified in the predefined class hierarchy using a data-driven approach; and it uses Gibbs Sampling to get the inferences out of the DN during the testing phase.

In one of the other two predictive performance measures, the PCT ensemble (PCTEN) algorithm was the best hierarchical classification algorithm in terms of average rank, but it was statistically significantly better than only two other classifiers. In the third predictive performance measure, the winner in terms of overall average rank was our proposed PCT-LHC hybrid algorithm, tied with the standard LHC algorithm. In this measure both algorithms were statistically significantly better than three other hierarchical classification algorithms.

8.1.4 Meta-Learning for Hierarchical Classification

We have proposed the first meta-learning approach for recommending the best hierarchical classification algorithm to a new hierarchical classification dataset. This resulted in three contributions, as follows.

First, we have proposed new meta-features for the meta-learning task of ranking hierarchical classification algorithms in new datasets. These new meta-features were created by using existing relations in the class hierarchy.

Second, we have developed a new algorithm to create new hierarchical datasets from existing ones based on the structure of the class hierarchy. This allowed us to substantially increase the number of hierarchical classification datasets used in our meta-learning experiments, addressing the main bottleneck of meta-learning experiments in general, which is the lack of availability of a large number of datasets.

Third, we have interpreted the meta-models to extract useful information about which hierarchical classification algorithm should be applied, depending on the characteristics of the hierarchical dataset to be analysed. Specifically, we have observed that the Decision Tree Meta Ranker (DTMR) method clearly outperformed the other three meta-rankers by a large margin. In addition, the DTMR method provided useful meta-knowledge from our meta-instances. This meta-knowledge can be broadly summarized as follows: the Ensemble of PCTs (PCTEN) hierarchical classification algorithm is more suited to relatively complex classification problems, the PCT algorithm is more suited to relatively simple problems with a reasonable number of features, the LHC algorithm performs better when the problem is relatively simple and has a reduced number of features and the HDN-nHPC algorithm is indicated when the landmark PCT model (a type of decision tree) for the hierarchical classification dataset (meta-instance) is large, which may indicates an over-fit of the PCT model.

8.1.5 Biological Interpretation of Classification Models in Ageing Research

We have performed an in-depth interpretation of PCT classification models induced to predict ageing-related protein functions with the assistance of the ageing biology expert Dr. Jennifer M. A. Tullet (University of Kent). In summary, we have concluded that the models have found (as relevant features) several protein-protein interactions involving genes that are known to be related to ageing. The models have also found protein-protein interactions in some model organisms that are known to be ageing related in other animals, which may indicate that the effect of the protein is conserved between species.

We have also discussed some elements of the classification model generated by the HDN-nHPC algorithm, showing that it was able to find strong relationships between ageing-related GO terms (class labels) that were not explicitly represented in the class hierarchy.

In addition, we have identified 14 papers containing the interpretation of some results obtained by supervised machine learning algorithms (i.e., classification or regression algorithms) applied to ageing research. We have observed that several key findings coming from biological ageing-research have been confirmed by supervised machine learning algorithms. For instance, it has been confirmed that ageing-related proteins tend to be more connected than non-ageing-related proteins, and several genes and proteins known to be involved in ageing were identified as ageing-related by data mining algorithms.

We have also observed that only one of the reviewed works performed wet-lab experimentation to validate the predictions of the supervised machine learning methods. This suggests that a greater integration between data mining and ageing experts would be a significant improvement in order to better translate the results of ageing research using machine learning (or data mining) methods into more established biological knowledge about ageing.

8.2 Future Work

Next, we list some possible lines of future work derived from this thesis.

A possible complementation of the study of the M-ELHNB algorithm may be the application of that algorithm in other domains. Another future work may be the extension of the algorithm for other types of hierarchical classification problems such as class structures organised as DAGs, instances annotated with multiple paths in the class hierarchy, and non-mandatory leaf class prediction classification problems.

Regarding the HDN-nHPC hierarchical classification algorithm, we plan to extract the strongest correlations found by the algorithm and analyse their biological meaning. As some correlations found in our data seem to be deterministic, (e.g.: some hierarchical classes never co-occur), this raises the question of whether or not these correlations should be explicitly represented in the class hierarchy, as making use of them could improve the predictive performance of other hierarchical classification algorithms used on this kind of data.

Also, the HDN-nHPC algorithm could be applied in the flat multi-label classification scenario, where there are no hierarchical relationships between classes. Flat multi-label problems are also common when classifying biological data.

A natural extension of the HDN-nHPC algorithm would be the application of some ensemble technique, such as *bagging* or *boosting*, to increase its predictive performance. It is well known that the use of ensemble techniques usually improves predictive performance of classification algorithms (Zhou 2012). One drawback of using such techniques is the increased running time. Actually, running the ensemble version of the HDN-nHPC algorithm on all datasets used in Chapter 6, with the available hardware, would require too much time (on the order of months). Therefore, we would need more processing power to run the current version of the HDN-nHPC algorithm in reasonable times, or we would need to implement optimisations and/or simplifications to drastically reduce the running time of the HDN-nHPC algorithm.

Another line of future research would be to improve the PCT-based hybrid algorithms in at least two ways: first, by studying the effects of using the HDN-nHPC algorithm in the hybrid (instead of using the HDN or LHC algorithms). Second, to devise a way to automatically select which hierarchical classification algorithm (among, e.g., HDN, LHC or HDN-nHPC) to use in the hybrid, based on the characteristics of the clusters formed by the PCT algorithm. The conclusions of our meta-learning study could be used here to help select which hierarchical classification algorithm to use.

Regarding meta-learning, we plan to test other types of kernel functions to try to improve the predictive performance of the meta-model generated by the Support Vector Machine (SVM) classification algorithm. Usually, SVMs tend to have better predictive performance than decision trees, which was the best meta-classification algorithm in terms of predictive power in our meta-learning study. We also plan to use other measures of performance that consider both the predictive performance of the methods and their running time.

One possible line of future research is to add more hierarchical classification algorithms to our meta-learning study, as well as further validation of our findings by adding to our experiments more hierarchical classification datasets that will be made available in the future by other researchers.

Finally, we plan to interact more with ageing specialists, further using data mining to try to discover new biological knowledge related to ageing. In particular, we plan to analyse other existing databases of ageing-related data, such as the Digital Ageing Atlas database (Craig et al. 2014), and discover new correlations between ageing and genes and/or protein functions.

Bibliography

Abdiansah, A. and Wardoyo, R. (2015). Time Complexity Analysis of Support Vector Machines (SVM) in LibSVM. *International Journal Computer and Application*, 128(3), pp. 28–34.

Abdulrahman, S. M. and Brazdil, P. (2014). Measures for combining accuracy and time for meta-learning. In *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection (MLAS'14)*, pp. 49–50.

Alaydie, N., Reddy, C. and Fotouhi, F. (2012). Exploiting Label Dependency for Hierarchical Multi-label Classification. In *Advances in Knowledge Discovery and Data Mining (PAKDD), Lecture Notes in Computer Science*, vol. 7301, Springer, pp. 294–305.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. (2008). *Molecular Biology of the Cell*. New York: Garland Science, 5th edn.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17), pp. 3389–3402.

Attwood, T. K., Bradley, P., Flower, D. R., Gaulton, A., Maudling, N., Mitchell, A. L., Moulton, G., Nordle, A., Paine, K., Taylor, P., Uddin, A. and Zygouri, C. (2003). PRINTS and its automatic supplement, prePRINTS. *Nucleic Acids Research*, 31(1), pp. 400–402.

Austad, S. (1997). Comparative Aging and Life Histories in Mammals. *Experimental Gerontology*, 32(1), pp. 23–38.

Bacardit, J. and Llorà, X. (2013). Large-scale data mining using genetics-based machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(1), pp. 37–61.

Bannister, L. A., Waldman, B. C. and Waldman, A. S. (2004). Modulation of error-prone double-strand break repair in mammalian chromosomes by DNA mismatch repair protein Mlh1. *DNA repair*, 3(5), pp. 465–474.

Barutcuoglu, Z. and DeCoro, C. (2006). Hierarchical Shape Classification Using Bayesian Aggregation. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pp. 44–48.

Barutcuoglu, Z., Schapire, R. E. and Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7), pp. 830–836.

Bedford, D. C. and Brindle, P. K. (2012). Is histone acetylation the most important physiological function for CBP and p300? *Aging*, 4(4), p. 247.

Bengio, Y., Delalleau, O. and Le Roux, N. (2005). The curse of dimensionality for local kernel machines. *Techn Rep*, 1258, Microsoft, `www.microsoft.com/en-us/research/publication/the-curse-of-dimensionality-for-local-kernel-machines/`.

Bhagwat, M. and Aravind, L. (2007). Psi-blast tutorial. In B. N. H, ed., *Comparative Genomics: Volumes 1 and 2*, Totowa (NJ): Humana Press, chap. 10.

Bi, W. and Kwok, J. T. (2011). Multi-Label Classification on Tree- and DAG-Structured Hierarchies. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pp. 17–24.

Blockeel, H., De Raedt, L. and Ramon, J. (1998). Top-down induction of clustering trees. *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, pp. 55–63.

Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J. and Struyf, J. (2002). Hierarchical Multi-Classification. In *Proceedings of the ACM SIGKDD 2002 workshop on multi-relational data mining (MRDM'02)*, pp. 1–15.

Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S. and Clare, A. (2006). Decision Trees for Hierarchical Multilabel Classification: A Case Study in Functional Genomics. In *Knowledge Discovery in Databases: PKDD 2006*, *Lecture Notes in Computer Science*, vol. 4213, Springer, pp. 18–29.

Bordes, A., Ertekin, S., Weston, J. and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep), pp. 1579–1619.

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT'92)*, ACM, pp. 144–152.

Boyd, K., Eng, K. H. and Page, C. D. (2013). Area Under the Precision-Recall Curve: Point Estimates and Confidence Intervals. In *Machine Learning and Knowledge Discovery in Databases.*, *Lecture Notes in Computer Science*, vol. 8190, Springer, pp. 451–466.

Bramer, M. (2013). *Principles of Data Mining*. Springer, 2nd edn.

Brazdil, P., Carrier, C. G., Soares, C. and Vilalta, R. (2008). *Metalearning: Applications to data mining*. Springer Science & Business Media.

Brazdil, P. B., Soares, C. and Da Costa, J. P. (2003). Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, 50(3), pp. 251–277.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), pp. 123–140.

Breitkreutz, D., Hlatky, L., Rietman, E. and Tuszynski, J. A. (2012). Molecular signaling network complexity is correlated with cancer patient survivability. *Proceedings of the National Academy of Sciences of the United States of America*, 109(23), pp. 9209–9212.

Brewer, G. J. (2007). Iron and copper toxicity in diseases of aging, particularly atherosclerosis and Alzheimers disease. *Experimental Biology and Medicine*, 232(2), pp. 323–335.

Brown-Borg, H. M., Kurt E. Borg, Charles J. Meliska and Bartke, A. (1996). Dwarf mice and the ageing process. *Nature*, 387(6604), p. 33.

Cai, C. (2003). SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Research*, 31(13), pp. 3692–3697.

Cai, L. and Hofmann, T. (2007). Exploiting known taxonomies in learning overlapping concepts. *Proceedings of the 2007 International Joint Conference on Artificial Intelligence (IJCAI'07)*, 7, pp. 714–719.

Campos, L. M. D., Fern, J. M. and Huete, J. F. (2006). A Theoretical Framework for Web Categorization in Hierarchical Directories using Bayesian Networks. *Soft Computing in Web Information Retrieval*, 197, pp. 25–43.

Cerri, R., Barros, R. C. and de Carvalho, A. C. (2014). Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1), pp. 39–56.

Cerri, R., Barros, R. C. and de Carvalho, A. C. P. L. F. (2011). Hierarchical Multi-Label Classification for Protein Function Prediction: A Local Approach Based on Neural Networks. In *Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA'11)*, pp. 337–343.

Cerri, R., Barros, R. C., de Carvalho, A. C. and Jin, Y. (2016). Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics*, 17(1), pp. 1–24.

Cesa-Bianchi, N., Gentile, C. and Zaniboni, L. (2006). Incremental Algorithms for Hierarchical Classification. *Journal of Machine Learning Research*, 7, pp. 31–54.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), pp. 1–27.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural computation*, 19(5), pp. 1155–1178.

Chapelle, O., Vapnik, V., Bousquet, O. and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1), pp. 131–159.

Charmpilas, N., Daskalaki, I., Papandreou, M. E. and Tavernarakis, N. (2014). Protein synthesis as an integral quality control mechanism during ageing. *Ageing research reviews*, 23(Part A), pp. 75–89.

Chen, L., Huang, T., Shi, X.-H., Cai, Y.-D. and Chou, K.-C. (2010). Analysis of protein pathway networks using hybrid properties. *Molecules*, 15(11), pp. 8177–8192.

Cheng, J. and Greiner, R. (2001). Learning Bayesian Belief Network Classifiers: Algorithms and System. In *Advances in Artificial Intelligence*, *Lecture Notes in Computer Science*, vol. 2056, Springer, pp. 141–151.

Claesen, M., De Smet, F., Suykens, J. A. and De Moor, B. (2014). Fast prediction with SVM models containing RBF kernels. *arXiv preprint arXiv:14030736.*

Clare, A. and King, R. D. (2003). Predicting gene function in Saccharomyces cerevisiae. *Bioinformatics*, 19(2), pp. ii42–ii49.

Comfort, A. (1964). *Ageing: the biology of senescence.* London: Routledge & Kegan Paul.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), pp. 273–297.

Craig, T., Smelick, C., Tacutu, R., Wuttke, D., Wood, S. H., Stanley, H., Janssens, G., Savitskaya, E., Moskalev, A., Arking, R. and de Magalhães, J. P. (2014). The digital ageing atlas: integrating the diversity of age-related changes into a unified resource. *Nucleic acids research*, 43(Database issue), pp. D873–D878.

Daitoku, H., Hatta, M., Matsuzaki, H., Aratani, S., Ohshima, T., Miyagishi, M., Nakajima, T. and Fukamizu, A. (2004). Silent information regulator 2 potentiates Foxo1-mediated transcription through its deacetylase activity. *Proceedings of the National Academy of Sciences of the United States of America*, 101(27), pp. 10042–10047.

Davis, J. and Goadrich, M. (2006). The Relationship Between Precision-Recall and ROC Curves. *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*, pp. 233–240.

de Magalhães, J. P. (2011). The biology of ageing: a primer. In *An Introduction to Gerontology*, Cambridge, UK: Cambridge University Press, chap. 2, 1st edn., pp. 22–47.

de Magalhães, J. P. (2014). Why genes extending lifespan in model organisms have not been consistently associated with human longevity and what it means to translation research. *Cell Cycle*, 13(17), pp. 2671–2673.

de Magalhães, J. P., Costa, J. and Church, G. M. (2007). An analysis of the relationship between metabolism, developmental schedules, and longenvity using phylogenetic independent contrasts. *Journal of Gerontology*, 2(62), pp. 149–160.

de Magalhães, J. P., Budovsky, A., Lehmann, G., Costa, J., Li, Y., Fraifeld, V. and Church, G. M. (2009). The Human Ageing Genomic Resources: online databases and tools for biogerontologists. *Aging Cell*, 8(1), pp. 65–72.

Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7(Jan), pp. 1–30.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* New Yotk: John Willey & Sons.

Dynan, W. S. and Yoo, S. (1998). Interaction of Ku protein and DNA-dependent protein kinase catalytic subunit with nucleic acids. *Nucleic acids research*, 26(7), pp. 1551–1559.

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS'01)*, pp. 681–687.

Eppig, J. T., Blake, J. A., Bult, C. J., Kadin, J. A., Richardson, J. E. and Mouse Genome Database Group (2015). The mouse genome database (MGD): facilitating mouse as a model for human biology and disease. *Nucleic Acids Research*, 43(D1), pp. D726–D736.

Fabris, F. and Freitas, A. A. (2014a). An Efficient Algorithm for Hierarchical Classification of Protein and Gene Functions. In *Proceedings of the 2014 25th International Workshop on Database and Expert Systems Applications (DEXA'14)*, pp. 64–68.

Fabris, F. and Freitas, A. A. (2014b). Dependency Network Methods for Hierarchical Multi-label Classification of Gene Functions. In *Proceedings of the 2014 IEEE International Conference on Computational Intelligence and Data Mining*, pp. 241–248.

Fabris, F. and Freitas, A. A. (2015). A Novel Extended Hierarchical Dependence Network Method Based on non-Hierarchical Predictive Classes and Applications to Ageing-Related Data. In *Proceedings of the 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI'15)*, IEEE, pp. 294–301.

Fabris, F. and Freitas, A. A. (2016). New KEGG pathway-based interpretable features for classifying ageing-related mouse proteins. *Bioinformatics*, 32(19), pp. 2988–2995.

Fabris, F., Freitas, A. A. and Magalhães, J. P. (2017). A review of supervised machine learning applied in ageing research (under review). *Biogerontology.*

Fabris, F., Freitas, A. A. and Tullet, J. (2015). An Extensive Empirical Comparison of Probabilistic Hierarchical Classifiers in Datasets of Ageing-Related Genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(6), pp. 1045–1058.

Fang, Y., Wang, X., Michaelis, E. K. and Fang, J. (2013). Classifying Aging Genes into DNA Repair or Non-DNA Repair-Related Categories. In *Proceedings of the 9th international conference on Intelligent Computing Theories and Technology (ICIC'13)*, *Lecture Notes in Computer Science*, vol. 7996, Springer, pp. 20–29.

Fattah, F., Lee, E. H., Weisensel, N., Wang, Y., Lichter, N. and Hendrickson, E. A. (2010). Ku regulates the non-homologous end joining pathway choice of DNA double-strand break repair in human somatic cells. *PLoS Genetics*, 6(2), pp. 1–14.

Feng, K., Song, X., Tan, F., Li, Y. H., Zhou, Y. C. and Li, J. H. (2012). Topological anaylysis and prediction of aging genes in Mus musculus. *2012 International Conference on Systems and Informatics (ICSAI'12)*, pp. 2268–2271.

Fernandes, M., Wan, C., Tacutu, R., Barardo, D., Rajput, A., Wang, J., Thoppil, H., Thornton, D., Yang, C., Freitas, A. and de Magalhães, J. P. (2017). Systematic analysis of the gerontome reveals links between aging and age-related diseases (in press). *Human Molecular Genetics.*

Filomeni, G., De Zio, D. and Cecconi, F. (2015). Oxidative stress and autophagy: the clash between damage and metabolic needs. *Cell Death & Differentiation*, 22(3), pp. 377–388.

Finn, R. D., Tate, J., Mistry, J., Coggill, P. C., Sammut, S. J., Hotz, H.-R., Ceric, G., Forslund, K., Eddy, S. R., Sonnhammer, E. L. L. and Bateman, A. (2008). The Pfam protein families database. *Nucleic Acids Research*, 36(suppl 1), pp. D281–D288.

Fortney, K., Kotlyar, M. and Jurisica, I. (2010). Inferring the functions of longevity genes with modular subnetwork biomarkers of Caenorhabditis elegans aging. *Genome biology*, 11(2), pp. 1–15.

Freitas, A. A. (2013). Comprehensible Classification Models - a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1), pp. 1–10.

Freitas, A. A. and de Carvalho, A. C. (2007). A Tutorial on Hierarchical Classification with Applications in Bioinformatics. In *Research and Trends in Data Mining Technologies and Applications*, IGI Global, chap. 7, pp. 175–208.

Freitas, A. A. and de Magalhães, J. P. (2011). A review and appraisal of the DNA damage theory of ageing. *Mutation Research - Reviews in Mutation Research*, 728(1-2), pp. 12–22.

Freitas, A. A., Vasieva, O. and de Magalhães, J. P. (2011). A data mining approach for classifying DNA repair genes into ageing-related or non-ageing-related. *BMC Genomics*, 1–11(1), p. 27.

Friedberg, I. (2006). Automated protein function prediction–the genomic challenge. *Briefings in Bioinformatics*, 7(3), pp. 225–242.

Friedman, D. and Johnson, T. (1988). A mutation in the age-1 gene in Caenorhabditis elegans lengthens life and reduces hermaphrodite fertility. *Genetics*, 118(1), pp. 75–86.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1), pp. 55–77.

Friedman, N., Geiger, D. and Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning*, 29(2), pp. 131–163.

Gámez, J., Mateo, J. and Puerta, J. (2008). Towards consistency in general dependency networks. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM'08)*, pp. 1–8.

Gámez, J. A., Mateo, J. L. and Puerta, J. M. (2006). Dependency networks based classifiers: learning models by using independence tests. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models (PGM'06)*, pp. 115–122.

Gámez, J. A., Mateo, J. L., Nielsen, T. D. and Puerta, J. M. (2008). Robust Classification using Mixtures of Dependency Networks. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM'08)*, pp. 129–136.

Gautheir, T. D. (2001). Detecting Trends Using Spearman's Rank Correlation Coefficient. *Environmental Forensics*, 2(4), pp. 359–362.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6), pp. 721–741.

Gems, D. and de la Guardia, Y. (2013). Alternative perspectives on aging in Caenorhabditis elegans: reactive oxygen species or hyperfunction? *Antioxidants & redox signaling*, 19(3), pp. 321–329.

Gjorgjevkikj, D., Madjarov, G. and Dzeroski, S. (2013). Hybrid Decision Tree Architecture Utilizing Local SVMs for Efficient Multi-Label Learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(7), pp. 1–12.

Goldman, D. P., Cutler, D., Rowe, J. W., Michaud, P.-C., Sullivan, J., Peneva, D. and Olshansky, S. J. (2013). Substantial health and economic returns from delayed aging may warrant a new focus for medical research. *Health Affairs*, 32(10), pp. 1698–1705.

Guevara-Aguirre, J., Balasubramanian, P., Guevara-Aguirre, M., Wei, M., Madia, F., Cheng, C.-W., Hwang, D., Martin-Montalvo, A., Saavedra, J., Ingles, S., de Cabo, R., Cohen, P. and Longo, V. D. (2011). Growth Hormone Receptor Deficiency Is Associated with a Major Reduction in Pro-Aging Signaling, Cancer, and Diabetes in Humans. *Science Translational Medicine*, 3(70), pp. 70ra13–70ra13.

Guo, Y. and Gu, S. (2011). Multi-Label Classification Using Conditional Dependency Networks. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, vol. 2, pp. 1300–1305.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations Newsletter*, 11(1), pp. 10–18.

Hall, M. A. (1999). *Correlation-based feature selection for machine learning.* Ph.D. thesis, The University of Waikato, New Zealand.

Hamza, M. and Larocque, D. (2005). An empirical comparison of ensemble methods based on classification trees. *Journal of Statistical Computation and Simulation*, 75(8), pp. 629–643.

Harman, D. (1956). Aging: a theory based on free radical and radiation chemistry. *Journal of Gerontology*, 3(11), pp. 298–300.

Harris, M. a., Clark, J., Ireland, A., Lomax, J., Ashburner, M., Foulger, R., Eilbeck, K., Lewis, S., Marshall, B., Mungall, C., Richter, J., Rubin, G. M., Blake, J. A., Bult, C., Dolan, M., Drabkin, H., Eppig, J. T., Hill, D. P., Ni, L., Ringwald, M., Balakrishnan, R., Cherry, J. M., Christie, K. R., Costanzo, M. C., Dwight, S. S., Engel, S., Fisk, D. G., Hirschman, J. E., Hong, E. L., Nash, R. S., Sethuraman, A., Theesfeld, C. L., Botstein, D., Dolinski, K., Feierbach, B., Berardini, T., Mundodi, S., Rhee, S. Y., Apweiler, R., Barrell, D., Camon, E., Dimmer, E., Lee, V., Chisholm, R., Gaudet, P., Kibbe, W., Kishore, R., Schwarz, E. M., Sternberg, P., Gwinn, M., Hannick, L., Wortman, J., Berriman, M., Wood, V., de la Cruz, N., Tonellato, P., Jaiswal, P., Seigfried, T. and White, R. (2004). The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32(Database issue), pp. D258–D261.

Hashimoto, H., Shintani, N. and Baba, A. (2002). Higher brain functions of PACAP and a homologous Drosophila memory gene amnesiac: insights from knockouts and mutants. *Biochemical and biophysical research communications*, 297(3), pp. 427–432.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R. and Kadie, C. (2001). Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research*, 1, pp. 49–75.

Hernandez, J., Sucar, L. and Morales, E. (2013). A Hybrid Global-Local Approach for Hierarchical Classification. In *Proceedings of the 26th Florida Artificial Intelligence Research Society International Conference (FLAIRS'13))*, pp. 432–437.

Hogeweg, P. (2011). The roots of bioinformatics in theoretical biology. *PLoS Computational Biology*, 7(3), pp. 1–5.

Holden, N. and Freitas, A. A. (2008). Improving the Performance of Hierarchical Classification with Swarm Intelligence. In *Proceedings of the 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO'08)*, pp. 48–60.

Hou, H.-A., Chou, W.-C., Lin, L.-I., Chen, C.-Y., Tang, J.-L., Tseng, M.-H., Huang, C.-F., Chiou, R.-J., Lee, F.-Y., Liu, M.-C. and Tien, H.-F. (2007). Characterization of acute myeloid leukemia with PTPN11 mutation: the mutation is closely associated with NPM1 mutation but inversely related to FLT3/ITD. *Leukemia*, 22(5), pp. 1075–1078.

Huang, T., Zhang, J., Xu, Z. P., Hu, L. L., Chen, L., Shao, J. L., Zhang, L., Kong, X. Y., Cai, Y. D. and Chou, K. C. (2012). Deciphering the effects of gene deletion on yeast longevity using network and machine learning approaches. *Biochimie*, 94(4), pp. 1017–1025.

Huerta, M., Downing, G., Haseltine, F., Seto, B. and Liu, Y. (2000). Nih working definition of bioinformatics and computational biology. *US National Institute of Health*.

Hunt, E. B., Marin, J. and Stone, P. J. (1966). Experiments in induction. *Academic Press*.

Hunter, S., Jones, P., Mitchell, A., Apweiler, R., Attwood, T. K., Bateman, A., Bernard, T., Binns, D., Bork, P., Burge, S., de Castro, E., Coggill, P., Corbett, M., Das, U., Daugherty, L., Duquenne, L., Finn, R. D., Fraser, M., Gough, J., Haft, D., Hulo, N., Kahn, D., Kelly, E., Letunic, I., Lonsdale, D., Lopez, R., Madera, M., Maslen, J., McAnulla, C., McDowall, J., McMenamin, C., Mi, H., Mutowo-Muellenet, P., Mulder, N., Natale, D., Orengo, C., Pesseat, S., Punta, M., Quinn, A. F., Rivoire, C., Sangrador-Vegas, A., Selengut, J. D., Sigrist, C. J. A., Scheremetjew, M., Tate, J., Thimmajanarthanan, M., Thomas, P. D., Wu, C. H., Yeats, C. and Yong, S.-Y. (2012). Interpro in 2011: new developments in the family and domain prediction database. *Nucleic Acids Research*, 40(D1), pp. D306–D312.

Jaber, N. and Zong, W.-X. (2013). Class III PI3K Vps34: essential roles in autophagy, endocytosis, and heart and liver function. *Annals of the New York Academy of Sciences*, 1280(1), pp. 48–51.

Japkowicz, N. and Shah, M. (2011). *Evaluating Learning Algorithms A Classification Perspective*. Cambridge, UK: Cambridge University Press, 1st edn.

Jiang, H. and Ching, W.-K. (2011). Classifying DNA repair genes by kernel-based support vector machines. *Bioinformation*, 7(5), pp. 257–263.

Jiang, Y., Oron, T., Clark, W. and et al. (2016). An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(184), pp. 1–70.

Jones, O. R., Scheuerlein, A., Salguero-Gómez, R., Camarda, C. G., Schaible, R., Casper, B. B., Dahlgren, J. P., Ehrlén, J., García, M. B., Menges, E. S., Quintana-Ascencio, P. F., Caswell, H., Baudisch, A. and Vaupel, J. W. (2014). Diversity of ageing across the tree of life. *Nature*, 505(7482), pp. 169–173.

Jungjit, S., Freitas, A. A., Michaelis, M. and Cinatl, J. (2014). Extending multi-label feature selection with kegg pathway information for microarray data analysis. In *Proceedings of the 2014 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'14)*, pp. 1–8.

Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M. and Tanabe, M. (2016). KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Research*, 44(D1), pp. D457–D462.

Kapahi, P., Chen, D., Rogers, A. N., Katewa, S. D., Li, P. W.-L., Thomas, E. L. and Kockel, L. (2010). With TOR, less is more: a key role for the conserved nutrient-sensing TOR pathway in aging. *Cell metabolism*, 11(6), pp. 453–465.

Karnoub, A. E. and Weinberg, R. A. (2008). Ras oncogenes: split personalities. *Nature reviews Molecular cell biology*, 9(7), pp. 517–531.

Keerthikumar, S., Bhadra, S., Kandasamy, K., Raju, R., Ramachandra, Y. L., Bhattacharyya, C., Imai, K., Ohara, O., Mohan, S. and Pandey, A. (2009). Prediction of candidate primary immunodeficiency disease genes using a support vector machine learning approach. *DNA Research*, 16(6), pp. 345–351.

Kenyon, C. J. (2010). The genetics of ageing. *Nature*, 464(7288), pp. 504–512.

Keogh, E. J. and Pazzani, M. J. (1999). Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceedings of the 7th International Workshop on AI and Statistics (AISTATS'99)*, pp. 1–6.

Kim, Y.-I., Cho, J. H., Yoo, O. J. and Ahnn, J. (2004). Transcriptional regulation and life-span modulation of cytosolic aconitase and ferritin genes in C. elegans. *Journal of molecular biology*, 342(2), pp. 421–433.

Kiritchenko, S., Matwin, S. and Famili, F. (2005). Functional Annotation of Genes Using Hierarchical Text Categorization. In *Proceedings of the Conference Linking Literature, Information and Knowledge for Biology (BioLINK SIG'05)*, pp. 1–4.

Kirkwood, T. B. L. (1977). Evolution of ageing. *Nature*, 270, pp. 301–304.

Kirkwood, T. B. L. (2002). Evolution of ageing. *Mechanisms of ageing and development*, 123(7), pp. 737–745.

Kirkwood, T. B. L. and Austad, S. N. (2000). Why do we age? *Nature*, 408, pp. 233–238.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models, Principles and Techniques*. MIT press, 1st edn.

Koller, D. and Sahami, M. (1997). Hierarchically Classifying Documents Using Very Few Words. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pp. 170–178.

Kurz, T., Terman, A. and Brunk, U. T. (2007). Autophagy, ageing and apoptosis: the role of oxidative stress and lysosomal iron. *Archives of Biochemistry and Biophysics*, 462(2), pp. 220–230.

Leite, R. and Brazdil, P. (2010). Active Testing Strategy to Predict the Best Classification Algorithm via Sampling and Meta-Learning. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*, pp. 309–314.

Leite, R., Brazdil, P. and Vanschoren, J. (2012). Selecting classification algorithms with active testing. In *Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Computer Science*, vol. 7376, pp. 117–131.

Li, Y. H., Dong, M. Q. and Guo, Z. (2010). Systematic analysis and prediction of longevity genes in Caenorhabditis elegans. *Mechanisms of Ageing and Development*, 131(11-12), pp. 700–709.

Li, Y. H., Zhang, G. G. and Guo, Z. (2010). Computational prediction of aging genes in human. *2010 International Conference on Biomedical Engineering and Computer Science (ICBECS'10)*, pp. 1–4.

Lichman, M. (2013). UCI machine learning repository archive.ics.uci.edu/ml.

Liu, G., Rogers, J., Murphy, C. T. and Rongo, C. (2011). EGF signalling activates the ubiquitin proteasome system to modulate C. elegans lifespan. *The EMBO journal*, 30(15), pp. 2990–3003.

Liu, H. and Motoda, H. (1998). *Feature extraction, construction and selection: A data mining perspective*. Springer Science & Business Media.

Lomax, R. G. and Hahs-Vaughn, D. L. (2013). *Statistical concepts: a second course*. Routledge.

Lombard, D. B., Beard, C., Johnson, B., Marciniak, R. A., Dausman, J., Bronson, R., Buhlmann, J. E., Lipman, R., Curry, R., Sharpe, A., Jaenisch, R. and Guarente, L. (2000). Mutations in the WRN gene in mice accelerate mortality in a p53-null background. *Molecular and Cellular Biology*, 20(9), pp. 3286–3291.

Lombard, D. B., Chua, K. F., Mostoslavsky, R., Franco, S., Gostissa, M. and Alt, F. W. (2005). DNA repair, genome stability, and aging. *Cell*, 120(4), pp. 497–512.

López-Otín, C., Blasco, M. A., Partridge, L., Serrano, M. and Kroemer, G. (2016). The Hallmarks of Aging. *Cell*, 153(6), pp. 1194–1217.

Loterman, G. and Mues, C. (2015). Learning algorithm selection for comprehensible regression analysis using datasetoids. *Intelligent Data Analysis*, 19(5), pp. 1019–1034.

Madden, T. (2013). *The BLAST sequence analysis tool*. National Center for Biotechnology Information (US).

Mair, W. and Dillin, A. (2008). Aging and survival: the genetics of life span extension by dietary restriction. *Annual Review of Biochemistry*, 77, pp. 727–754.

Mayne, A. and Perry, R. (2009). Hierarchically classifying documents with multiple labels. In *Proceedings of the IEEE 2009 Symposium on Computational Intelligence and Data Mining (CIDM'09)*, pp. 133–139.

The Uniprot Consortium (2007). The Universal Protein Resource (UniProt). *Nucleic Acids Research*, 35(Database issue), pp. D193–D197.

The Uniprot Consortium (2010). The Universal Protein Resource (UniProt) in 2010. *Nucleic Acids Research*, 38(Database issue), pp. D142–D148.

Medvedev, Z. A. (1990). An Attempt at a Rational Classification of Theories of Ageing. *Biological Review*, 65(3), pp. 375–398.

Merschmann, L. d. C. and Freitas, A. A. (2013). An Extended Local Hierarchical Classifier for Prediction of Protein and Gene Functions. In *Proceedings of the 15th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'13)*, *Lecture Notes in Computer Science*, vol. 8057, pp. 159–171.

Nakamura, E. and Miyao, K. (2007). A method for identifying biomarkers of aging and constructing an index of biological age in humans. *Journal of Gerontology*, 62(10), pp. 1096–1105.

Neville, J. and Jensen, D. (2007). Relational Dependency Networks. *Journal of Machine Learning Research*, 8, pp. 653–692.

Norgate, M., Lee, E., Southon, A., Farlow, A., Batterham, P., Camakaris, J. and Burke, R. (2006). Essential roles in development and pigmentation for the Drosophila copper transporter DmATP7. *Molecular biology of the cell*, 17(1), pp. 475–484.

Obozinski, G., Lanckriet, G. and Grant, C. (2008). Consistent probabilistic outputs for protein function prediction. *Genome Biology*, 9(1), pp. 1–19.

Okuyama, T., Inoue, H., Ookuma, S., Satoh, T., Kano, K., Honjoh, S., Hisamoto, N., Matsumoto, K. and Nishida, E. (2010). The ERK-MAPK pathway regulates longevity through SKN-1 and insulin-like signaling in Caenorhabditis elegans. *Journal of Biological Chemistry*, 285(39), pp. 30274–30281.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11, pp. 169–198.

Otero, F. E. B., Freitas, A. A. and Johnson, C. G. (2010). A Hierarchical Multi-Label Classification Ant Colony Algorithm for Protein Function Prediction. *Memetic Computing*, 2(3), pp. 165–181.

Pandey, G., Kumar, V. and Steinbach, M. (2006). Computational approaches for protein function prediction. Tech. Rep. TR 06-028, Department of Computer Science and Engineering, University of Minnesota, Twin Cities.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Peng, Y., Flach, P. A., Soares, C. and Brazdil, P. B. (2002). Improved dataset characterisation for meta-learning. In *Discovery Science*, *Lecture Notes in Computer Science*, vol. 2534, pp. 141–152.

Pugelj, M. and Dzeroski, S. (2011). Predicting Structured Outputs k-Nearest Neighbours Method. In *Discovery Science*, *Lecture Notes in Computer Science*, vol. 6926, pp. 262–276.

Putin, E., Mamoshina, P., Aliper, A., Korzinkin, M. and Moskalev, A. (2016). Deep biomarkers of human aging : Application of deep neural networks to biomarker development. *Aging*, 8(5), pp. 1–13.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ramírez-Corona, M., Sucar, L. E. and Morales, E. F. (2016). Hierarchical multilabel classification based on path evaluation. *International Journal of Approximate Reasoning*, 68, pp. 179–193.

Rokach, L. and Maimon, O. (2005). Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4), pp. 476–487.

Rousu, J. and Shawe-Taylor, J. (2006). Kernel-Based Learning of Hierarchical Multilabel Classification Models. *Journal of Machine Learning Research*, 7, pp. 1601–1626.

Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Guldener, U., Mannhaupt, G., Mnsterktter, M. and Mewes, H. W. (2004). The

FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18), pp. 5539–5545.

Salama, K. M. and Freitas, A. A. (2013). ACO-Based Bayesian Network Ensembles for the Hierarchical Classification of Ageing-Related Proteins. In *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, *Lecture Notes in Computer Science*, vol. 7833, pp. 80–91.

Sandberg, M., Eriksson, L., Jonsson, J., Sjöström, M. and Wold, S. (1998). New chemical descriptors relevant for the design of biologically active peptides. A multivariate characterization of 87 amino acids. *Journal of Medicinal Chemistry*, 41(14), pp. 2481–2491.

Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D. and Dzeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11(2), pp. 1–14.

Secker, A., Davies, M. N., Freitas, A. A., Timmis, J., Mendao, M. and Flower, D. R. (2007). An Experimental Comparison of Classification Algorithms for the Hierarchical Prediction of Protein Function Classification of GPCRs. *Expert Update*, 9(3), pp. 17–22.

Sen, P., Namata, G., Bilgic, M. and Getoor, L. (2008). Collective Classification in Network Data. *AI magazine*, 29(3), pp. 1–24.

Shaposhnikov, M., Proshkina, E., Shilova, L., Zhavoronkov, A. and Moskalev, A. (2015). Lifespan and stress resistance in Drosophila with overexpressed DNA repair genes. *Scientific reports*, 5, pp. 1–12.

Sharan, R., Ulitsky, I. and Shamir, R. (2007). Network-based prediction of protein function. *Molecular Systems Biology*, 3(1), pp. 1–13.

Sigrist, C. J. A., de Castro, E., Cerutti, L., Cuche, B. A., Hulo, N., Bridge, A., Bougueleret, L. and Xenarios, I. (2013). New and continuing developments at PROSITE. *Nucleic Acids Research*, 41(D1), pp. D344–D347.

Silla Jr., C. N. and Freitas, A. A. (2009). A Global-Model Naive Bayes Approach to the Hierarchical Prediction of Protein Functions. In *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'09)*, pp. 992–997.

Silla Jr., C. N. and Freitas, A. A. (2009). Novel Top-Down Approaches for Hierarchical Classification and Their Application to Automatic Music Genre Classification. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'09)*, pp. 3499–3504.

Silla Jr., C. N. and Freitas, A. A. (2011a). A Survey of Hierarchical Classification Across Different Application Domains. *Data Mining and Knowledge Discovery*, 44(1-2), pp. 31–72.

Silla Jr., C. N. and Freitas, A. A. (2011b). Selecting different protein representations and classification algorithms in hierarchical protein function prediction. *Intelligent Data Analysis*, 15(6), pp. 979–999.

Song, X., Zhou, Y.-C., Feng, K., Li, Y.-H. and Li, J.-H. (2012). Discovering aging-genes by topological features in Drosophila melanogaster protein-protein interaction network. In *2012 IEEE 12th International Conference on Data Mining Workshops (ICDM'12)*, IEEE, pp. 94–98.

Stein, L. D. (2003). Integrating biological databases. *Nature reviews Genetics*, 4(5), pp. 337–345.

Sun, Q. and Pfahringer, B. (2013). Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine Learning*, 93(1), pp. 141–161.

Swan, A. L., Mobasheri, A., Allaway, D., Liddell, S. and Bacardit, J. (2013). Application of machine learning to proteomics data: classification and biomarker identification in postgenomics biology. *Omics: a journal of integrative biology*, 17(12), pp. 595–610.

Szilard, L. (1959). On the nature of the aging process. *Proceedings of the National Academy of Sciences of the United States of America*, 1(45), pp. 30–45.

Tacutu, R., Craig, T., Budovsky, A., Wuttke, D., Lehmann, G., Taranukha, D., Costa, J., Fraifeld, V. E. and de Magalhães, J. P. (2013). Human Ageing Genomic Resources: integrated databases and tools for the biology and genetics of ageing. *Nucleic Acids Research*, 41(Database issue), pp. D1027–D1033.

Tamura, T., Chiang, A.-S., Ito, N., Liu, H.-P., Horiuchi, J., Tully, T. and Saitoe, M. (2003). Aging specifically impairs amnesiac-dependent memory in Drosophila. *Neuron*, 40(5), pp. 1003–1011.

Tan, P.-N., Steinbach, M. and Kumar, V. (2006). *Introduction to data mining.* Pearson.

The UniProt Consortium (2014). Uniprot: a hub for protein information. *Nucleic acids research*, 43(Database issue), pp. D204–D212.

Tian, Y., Yang, Q., Huang, T., Ling, C. X. and Gao, W. (2006). Learning Contextual Dependency Network Models for Link-Based Classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11), pp. 1482–1496.

Tidyman, W. E. and Rauen, K. A. (2009). The RASopathies: developmental syndromes of Ras/MAPK pathway dysregulation. *Current opinion in genetics & development*, 19(3), pp. 230–236.

Toutanova, K. and Klein, D. (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL'03)*, vol. 1, pp. 173–180.

Tsoumakas, G., Katakis, I. and Vlahavas, I. (2010). Mining Multi-label Data. In *Data Mining and Knowledge Discovery Handbook*, Springer US, pp. 667–685.

Udell, C. M., Rajakulendran, T., Sicheri, F. and Therrien, M. (2011). Mechanistic principles of RAF kinase signaling. *Cellular and Molecular Life Sciences*, 68(4), pp. 553–565.

Valentini, G. (2009). True Path Rule Hierarchical Ensembles. In *Multiple Classifier Systems*, *Lecture Notes in Computer Science*, vol. 5519, pp. 232–241.

Valentini, G. (2011). True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3), pp. 832–847.

Valentini, G. and Cesa-Bianchi, N. (2008). HCGene: a software tool to support the hierarchical classification of genes. *Bioinformatics*, 24(5), pp. 729–731.

Valentini, G. and Re, M. (2009). Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction. In *Proceedings of the 1st workshop on learning from multi-label data (MLD) held in conjunction with ECML/PKDD'09*, pp. 132–145.

van Heemst, D., Mooijaart, S. P., Beekman, M., Schreuder, J., de Craen, A. J., Brandt, B. W., Eline Slagboom, P. and Westendorp, R. G. (2005). Variation in the human TP53 gene affects old age survival and cancer mortality. *Experimental gerontology*, 40(1), pp. 11–15.

van Rijn, J. N., Abdulrahman, S. M., Brazdil, P. and Vanschoren, J. (2015). Fast algorithm selection using learning curves. In *Proceedings of the 2015 International Symposium on Intelligent Data Analysis (IDA'15)*, pp. 298–309.

Vens, C., Struyf, J., Schietgat, L., Dzeroski, S. and Blockeel, H. (2008). Decision Trees for Hierarchical Multi-label Classification. *Machine Learning*, 73(2), pp. 185–214.

Vens, C., Schietgat, L., Struyf, J., Blockeel, H. and Kocev, D. (2010). Predicting Gene Function using Predictive Clustering Trees. *BMC Bioinformatics*, 11(2), pp. 1–25.

Wan, C. and Freitas, A. (2013). Prediction of the pro-longevity or anti-longevity effect of Caenorhabditis Elegans genes based on Bayesian classification methods. In *Proceedings of the 2013 IEEE International Conference on Bioinformatics and Biomedicine (BIBM'13)*, pp. 373–380.

Wan, C., Freitas, A. A. and de Magalhães, J. P. (2015). Predicting the pro-longevity or anti-longevity effect of model organism genes with new hierarchical feature selection methods. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(2), pp. 262–275.

Wang, K., Zhou, S. and He, Y. (2001). Hierarchical classification of real life documents. In *Proceedings of the 2001 International Conference on Data Mining (ICDM'01)*, pp. 1–16.

Watson, J. D. and Crick, F. H. C. (1953). Molecular structure of nucleic acid. *Nature*, (171), pp. 737–738.

Wieser, D., Papatheodorou, I., Ziehm, M. and Thornton, J. M. (2011). Computational biology for ageing. *Philosophical transactions of the Royal Society of London Series B, Biological sciences*, 366(1561), pp. 51–63.

Williams, G. C. (1957). Pleitropy, Natural Selection, and the Evolution of Senescence. *Evolution*, 11(4), pp. 398–411.

Witten, I. H. and Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Xia, J. and Wishart, D. S. (2010). MetPA: a web-based metabolomics tool for pathway analysis and visualization. *Bioinformatics*, 26(18), pp. 2342–2344.

Xiao, Z., Dellandrea, E., Dou, W. and Chen, L. (2009). Multi-stage classification of emotional speech motivated by a dimensional emotion model. *Multimedia Tools and Applications*, 46(1), pp. 119–145.

Xiao, Z.-Q., Yu, Y., Khan, A., Jaszewski, R., Ehrinpreis, M. N. and Majumdar, A. P. (1999). Induction of G1 checkpoint in the gastric mucosa of aged rats. *American Journal of Physiology-Gastrointestinal and Liver Physiology*, 277(5), pp. G929–G934.

Xu, J., Liu, D. and Songyang, Z. (2002). The role of Asp-462 in regulating Akt activity. *Journal of Biological Chemistry*, 277(38), pp. 35561–35566.

Yang, Z. and Bielawski, J. P. (2000). Statistical methods for detecting molecular adaptation. *Trends in Ecology & Evolution*, 15(12), pp. 496–503.

Zhang, J. D. and Wiemann, S. (2009). KEGGgraph: a graph approach to KEGG PATHWAY in R and bioconductor. *Bioinformatics*, 25(11), pp. 1470–1471.

Zheng, H., Li, S., Hsu, P. and Qu, C.-K. (2013). Induction of a tumor-associated activating mutation in protein tyrosine phosphatase Ptpn11 (Shp2) enhances mitochondrial metabolism, leading to oxidative stress and senescence. *Journal of Biological Chemistry*, 288(36), pp. 25727–25738.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.